Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 326 (2004) 343–362
Theoretical Computer Science
www.elsevier.com/locate/tcs

Optimal graph exploration without good maps[☆]

Anders Dessmark[a,1], Andrzej Pelc[b,*,2]

[a] *Department of Computer Science, Lund Institute of Technology, Box 118, S-22100 Lund, Sweden*
[b] *Département d'informatique, Université du Québec en Outaouais, Hull, Québec J8X 3X7, Canada*

Received 20 October 2002; received in revised form 29 June 2004; accepted 19 July 2004
Communicated by D. Peleg

Abstract

A robot has to visit all nodes and traverse all edges of an unknown undirected connected graph, using as few edge traversals as possible. The quality of an exploration algorithm $\mathscr{A}$ is measured by comparing its cost (number of edge traversals) to that of the optimal algorithm having full knowledge of the graph. The ratio between these costs, maximized over all starting nodes in the graph and over all graphs in a given class $\mathscr{U}$, is called the *overhead* of algorithm $\mathscr{A}$ for the class $\mathscr{U}$ of graphs. We consider three scenarios, providing the robot with varying amount of information. The robot may either know nothing about the explored graph, or have an unlabeled isomorphic copy of it (an *unanchored map*), or have such a copy with a marked starting node (an *anchored map*).

For all of the above scenarios, we construct natural exploration algorithms that have smallest, or—in one case—close to smallest, overhead. While for the class of all graphs, depth-first search turns out to be an optimal algorithm for all scenarios, the situation for trees is much different. We show that, under the scenario without any knowledge, DFS is still optimal for trees but this is not the case if a map is available. Under the scenario with an unanchored map, we show that optimal overhead is at least $\sqrt{3}$ but strictly below 2 (and thus DFS is not optimal). Under the scenario with an anchored map, we construct an optimal algorithm for trees and show that its overhead is $\frac{3}{2}$. We also consider exploration of the class of lines (simple paths). In this case, depth-first search remains optimal for the scenario without any knowledge, with overhead 2. Under the scenario with an unanchored map, we construct an optimal algorithm and show that its overhead is $\sqrt{3}$. Finally, under the scenario with an anchored

[☆] A preliminary version of this paper appeared at the 10th European Symposium on Algorithms (ESA 2002).

[*] Corresponding author.
E-mail addresses: andersd@cs.lth.se (A. Dessmark), pelc@uqo.ca (A. Pelc).

[1] This research was done during a visit of Anders Dessmark at the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

[2] Andrzej Pelc was supported in part by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.07.031

map, we construct an optimal algorithm and show that its overhead is $\frac{7}{5}$. An important contribution of this paper is establishing lower bounds that prove optimality of these exploration algorithms.

Keywords: Algorithm; Exploration; Graph; Map; Robot; Traversal

1. Introduction

A robot has to visit all nodes and traverse all edges of an unknown undirected connected graph, using as few edge traversals as possible. If the robot has complete knowledge of the explored graph G, i.e., if it has an oriented labeled isomorphic copy of it showing which port at a visited node leads to which neighbor, then exploration with fewest edge traversals starting from node v corresponds to the shortest *covering walk* from v: the shortest, not necessarily simple, path in G starting from v and containing all edges. The length of this shortest covering walk is called the *cost* of G from v, and is denoted $opt(G, v)$. For example, if G is an Eulerian graph then $opt(G, v)$ is the number of edges in G, for any v, and if G is a tree then $opt(G, v) = 2(n - 1) - ecc(v)$, where n is the number of nodes in G and $ecc(v)$ is the eccentricity of the starting node v, i.e., the distance from v to the farthest leaf. In this latter case, depth-first search ending in the leaf farthest from the starting node v clearly uses fewest edge traversals.

However, graph exploration is often performed when the explored graph is partially or totally unknown. We consider three scenarios, providing the robot with varying amount of information. Under the first scenario, the robot does not have any a priori knowledge of the explored graph. We refer to this scenario as *exploration without a map*. Under the second scenario, the robot has an unlabeled isomorphic copy of the explored graph. We call it an *unanchored map* of the graph. Finally, under the third scenario, the robot has an unlabeled isomorphic copy of the explored graph with a marked starting node. We call it an *anchored map* of the graph. It should be stressed that even the scenario with an anchored map does not give the robot any sense of direction, since the map is unlabeled. For example, when the robot starts the exploration of a line, such a map gives information about the length of the line and distances from the starting node to both ends, but does not tell which way is the closest end. In the case of an $n \times m$ torus, the availability of either type of map is equivalent to the information that the explored graph is an $n \times m$ torus.

In all scenarios we assume that all nodes have distinct labels, and all ports at a node v are numbered $1, \ldots, \deg(v)$ (in the explored graph, not in the map). Hence the robot can recognize already visited nodes and traversed edges. However, it cannot tell the difference between yet unexplored edges incident to its current position, i.e., it does not know the other ends of such edges. If the robot decides to use such an unexplored edge, the actual choice of the edge belongs to the adversary, as we are interested in worst-case performance. For a given exploration algorithm $\mathcal{A}$, the *cost* $\mathcal{C}(\mathcal{A}, G, v)$ of this algorithm run on a graph G from a starting node v is the worst-case number of edge traversals taken over all of the above choices of the adversary.

For a given graph G and a given starting node v, a natural measure of quality of an exploration algorithm $\mathcal{A}$ is the ratio $\mathcal{C}(\mathcal{A}, G, v)/opt(G, v)$ of its cost to that of the optimal algorithm having complete knowledge of the graph. This ratio represents the relative penalty payed by the algorithm for the lack of knowledge of the environment. For a given class $\mathcal{U}$ of graphs, the number

$$\mathcal{O}_{\mathcal{U}}(\mathcal{A}) = \sup_{G \in \mathcal{U}} \max_{v \in G} \frac{\mathcal{C}(\mathcal{A}, G, v)}{opt(G, v)}$$

is called the *overhead* of algorithm $\mathcal{A}$ for the class $\mathcal{U}$ of graphs. It is the maximum relative penalty described above, over all starting nodes in all graphs of the class. The lower the overhead of an exploration algorithm, the closer is its performance (in the worst case) to that of the optimal algorithm having full knowledge of the environment. For a fixed scenario, an algorithm is called *optimal* for a given class of graphs, if its overhead for this class is minimal among all exploration algorithms working under this scenario.

Since $\mathcal{C}(DFS, G, v) \leqslant 2e$, and $opt(G, v) \geqslant e$, for any graph G with e edges and any starting node v (depth-first search traverses each edge at most twice, and every edge has to be traversed at least once), it follows that the overhead of DFS is at most 2, for any class of graphs. Hence, for any class of graphs, the overhead of an optimal algorithm is between 1 and 2, under every scenario (DFS does not use any information about the explored graph).

The following remark will be useful for proving lower bounds on overhead of exploration algorithms. Suppose that the robot, at some point of the exploration, is at node w, then moves along an already explored edge e incident to w, and immediately returns to w. For any set of decisions of the adversary, an algorithm causing such a pair of moves, when run on a graph G from some starting node v, has cost strictly larger than the algorithm that skips these two moves. Hence, we restrict attention to exploration algorithms that never perform such returns. We call them *regular*.

1.1. Related work

Exploration and navigation problems for robots in an unknown environment have been extensively studied in the literature (cf. the survey [19]). There are two principal ways of modeling the explored environment. In one of them a geometric setting is assumed, e.g., unknown terrain with convex obstacles [8], or room with polygonal [9] or rectangular [3] obstacles. Another way is to represent the unknown environment as a graph, assuming that the robot may only move along its edges. The graph model can be further specified in two different ways. In [1,4,5,11] the robot explores strongly connected directed graphs and it can move only in the direction from head to tail of an edge, not vice-versa. In [2,7,15–17] the explored graph is undirected and the robot can traverse edges in both directions. The efficiency measure adopted in most papers dealing with exploration of graphs is the cost of completing this task, measured by the number of edge traversals by the robot. In some papers additional restrictions on the moves of the robot are imposed. It is assumed that the robot has either a restricted tank [2,7], forcing it to periodically return to the base for refueling, or that it is tethered, i.e., attached to the base by a rope or cable of restricted length [15]. It is proved in [15] that exploration can be done in time $O(e)$ under both scenarios. Another direction of research concerns exploration of anonymous graphs. In this case it is impossible to explore arbitrary graphs if no marking of nodes is allowed. Hence the scenario adopted

Table 1
Summary of results

	Anchored map	Unanchored map	No map
Lines	Overhead: $\frac{7}{5}$ optimal	Overhead: $\sqrt{3}$ optimal	Depth-first search
Trees	Overhead: $\frac{3}{2}$ optimal	Overhead: < 2 lower bound $\sqrt{3}$	overhead: 2
General graphs	Depth-first search, overhead: 2, optimal		

in [4,5] is to allow *pebbles* which the robot can drop on nodes to recognize already visited ones, and then remove them and drop in other places. The authors concentrate attention on the minimum number of pebbles allowing efficient exploration of arbitrary directed graphs. Exploring anonymous undirected trees without the possibility of marking nodes is investigated in [12]. The authors concentrate attention not on the cost of exploration but on the minimum amount of memory sufficient to carry out this task. Exploration of anonymous graphs was also considered in [10,13,14].

The work most closely related to the present paper is that from [17]. The authors consider exploration of undirected graphs (both arbitrary graphs and trees). The adopted efficiency measure is similar in spirit to our notion of overhead but differs from it in an important way. Similarly as in the present paper, in [17] the authors consider the ratio of the cost of an algorithm lacking some knowledge of the graph to that of the optimal algorithm having this knowledge. (In particular, they study the scenario with an unanchored map.) However, for a given graph, both costs are maximized over all starting nodes, and the ratio of these maxima is considered as the performance measure of the algorithm on the graph. (Then the supremum of this ratio is taken over all graphs in the considered class). This approach should be contrasted with our definition of overhead, where the ratio is computed for each starting node individually and then maximized over all possible starting nodes in the graph. In order to see the difference between both approaches, consider the case of the line with availability of an unanchored map (which, for the line, is equivalent to knowing its length). The maximum cost of depth-first search on the line L_n of length n is $2n-1$ (the maximum taken over all starting nodes). On the other hand, $opt(L_n, v) \geqslant 3n/2-2$ for *some* starting node v. This gives a ratio close to $\frac{4}{3}$ and leads to the conclusion, proved in [17], that DFS is optimal for lines (and in fact for all trees), according to their measure. However, this measure (and hence the obtained result) can be viewed as biased in favor of DFS because for some starting nodes v (close to the endpoints of the line) the ratio of the cost of DFS to $opt(L_n, v)$ is approximately 2. This is captured by our notion of overhead, and in fact leads to the conclusion that, if n is known, there are exploration algorithms of a line better than DFS, according to the overhead measure (Table 1).

1.2. Our results

The aim of this paper is to establish which exploration algorithms have the lowest possible overhead for each of the three scenarios described above: no knowledge of the graph, availability of an unanchored map, and availability of an anchored map. It turns out that some of these algorithms are fairly natural and our main contribution is proving lower bounds that show their optimality.

Depth-first search is among the most natural exploration algorithms in an unknown graph. At every point of the exploration the robot chooses an unexplored edge, if it exists. Otherwise, it backtracks to the most recently visited node with an unexplored incident edge. If no such node exists, exploration is completed. Since DFS traverses every edge at most twice, regardless of adversary's choices, its overhead is at most 2, for all classes of graphs.

While for the class of all (undirected, connected) graphs, depth-first search turns out to be an optimal algorithm for all scenarios, the situation for trees is much different. We show that, under the scenario without any knowledge, DFS is still optimal for trees but this is not the case if a map is available. Under the scenario with an unanchored map, we show that optimal overhead is at least $\sqrt{3}$ but strictly below 2 (and thus DFS, with overhead 2, is not optimal). Under the scenario with an anchored map, we construct an optimal algorithm for trees and show that its overhead is $\frac{3}{2}$. We also consider exploration of the class of lines (simple paths). In this case, depth-first search remains optimal for the scenario without any knowledge, with overhead 2. Under the scenario with an unanchored map, we construct an optimal algorithm and show that its overhead is $\sqrt{3}$. Finally, under the scenario with an anchored map, we construct an optimal algorithm and show that its overhead is $\frac{7}{5}$. A summary of our results is contained in Table 1.

The paper is organized as follows. In Section 2 we consider the class of lines, and construct for it optimal exploration algorithms under all three scenarios. In Section 3 we consider arbitrary trees: for the scenario with an anchored map we show an optimal exploration algorithm, and for the scenario with an unanchored map we give estimates on the overhead of optimal exploration. (For the scenario without any knowledge of the graph, optimality of DFS follows from Section 2). Finally, in Section 4 we show that any exploration algorithm has overhead at least 2 for the class of arbitrary graphs, even when an anchored map is available, and hence DFS is optimal for this class, under all three scenarios.

2. Lines

In this section we construct optimal exploration algorithms for the class $\mathcal{L}$ of lines. A line of length n is a graph $L_n = (V, E)$, where $V = \{v_0, \ldots, v_n\}$ and $E = \{[v_i, v_{i+1}] : i = 0, 1, \ldots, n-1\}$. It turns out that, for all three scenarios, optimal algorithms require at most 2 returns (changes of direction) on the line.

2.1. Exploration with an anchored map

We consider the scenario in which an anchored map of the line is available to the robot. This is equivalent to knowing the length n of the line and the distances a and b between the starting node and the endpoints. Assume that $a \leqslant b$. We describe an exploration algorithm, establish its overhead, and prove that it is optimal.

```
Algorithm Anchored-Line
• Let $x = 3a + n$ and $y = 2n - a$.
• If $x \leqslant y$ then
    ◦ go at distance $a$ in one direction, or until an endpoint is reached, whichever comes
      first;
    ◦ if an endpoint is reached then
            return, go to the other endpoint, and stop
      else
            return, go to the endpoint, return, go to the other endpoint, and stop
  else
    ◦ go to the endpoint in one direction, return, go to the other endpoint, and stop.
```

Theorem 2.1. *Algorithm Anchored-Line has overhead $\frac{7}{5}$ for the class $\mathcal{L}$ of lines.*

Proof. Denote Algorithm Anchored-Line by $\mathcal{A}$. By definition, $\mathcal{C}(\mathcal{A}, L_n, v) = \min(x, y)$. Since $opt(L_n, v) = a + n$, we have:
if $a \leqslant \lfloor n/4 \rfloor$ then $\mathcal{C}(\mathcal{A}, L_n, v)/opt(L_n, v) \leqslant \frac{3\lfloor n/4 \rfloor + n}{\lfloor n/4 \rfloor + n} \leqslant \frac{7}{5}$;
if $a \geqslant \lceil n/4 \rceil$ then $\mathcal{C}(\mathcal{A}, L_n, v)/opt(L_n, v) \leqslant \frac{2n - \lceil n/4 \rceil}{\lceil n/4 \rceil + n} \leqslant \frac{7}{5}$. Hence $\mathcal{O}_{\mathcal{L}}(\mathcal{A}) \leqslant 7/5$. Since, for an arbitrary n divisible by 4 and $a = n/4$, we have $\mathcal{C}(\mathcal{A}, L_n, v)/opt(L_n, v) = \frac{7}{5}$, this proves $\mathcal{O}_{\mathcal{L}}(\mathcal{A}) = \frac{7}{5}$. □

The next theorem proves that Algorithm Anchored-Line is optimal for the class of lines.

Theorem 2.2. *Every exploration algorithm with an anchored map has overhead at least $\frac{7}{5}$ for the class of lines.*

Proof. Consider any (regular) exploration algorithm $\mathcal{E}$. If the robot always starts the exploration by going in one direction till the endpoint (in this case $\mathcal{E}$ is simply DFS), then its overhead is 2 as witnessed by the starting node for which $a = 1$, considered for lines of all possible lengths. Otherwise, let c be the number of steps in one direction after which $\mathcal{E}$ returns if it has not reached an endpoint. (c can depend on n and a). If $c < a$ then the robot goes c steps and then back to the starting node (by regularity), regardless of the direction chosen by the adversary. The algorithm skipping these first $2c$ steps and then following $\mathcal{E}$ has strictly smaller cost. Suppose that $c > a$. In some cases the robot does not reach the endpoint in the first chosen direction (otherwise $\mathcal{E}$ would be DFS, which was already analyzed). Hence c must be smaller than b. In this case, the algorithm in which the robot goes a step in one direction (instead of c steps), and then returns if it has not reached an endpoint, subsequently behaving like $\mathcal{E}$, has strictly smaller cost than $\mathcal{E}$, regardless of the direction chosen by the adversary. It follows that in order to prove the lower bound, it is enough to restrict attention to algorithms for which $c = a$. Suppose that the direction chosen by the adversary is towards the farthest endpoint. Then the robot is back at the starting node after making $2a$ steps and not visiting any endpoint. Consequently the robot has to make at least $a + n$ further steps (the value of $opt(L_n, v)$) to finish exploration. This implies that the cost of the algorithm is at least $3a + n$, and hence its overhead is not smaller than that of Algorithm Anchored-Line. □

2.2. Exploration with an unanchored map

We now present an algorithm for exploration of lines with an unanchored map (i.e., the length n of the line L_n is known to the robot but the starting node v is unknown) that has overhead $\sqrt{3}$, and show that it is optimal for the class of lines.

Algorithm Unanchored-Line

```
• Let $a = \lfloor \frac{\sqrt{3}-1}{2} n \rfloor$.
• If the starting node $v$ is an endpoint then
    ◦ go to the other endpoint and stop.
  else
    ◦ go at distance $a$ in one direction or until an endpoint is reached, whichever comes
      first.
    ◦ if an endpoint is reached then
              return, go to the other endpoint, and stop
      else
              return, go to the endpoint, return, go to the other endpoint, and stop.
```

Theorem 2.3. *Algorithm Unanchored-Line has overhead not larger than $\sqrt{3}$ for the class $\mathcal{L}$ of lines.*

Proof. Let x be the distance between v and the endpoint of L_n that is not in the direction chosen by the adversary for the first traversal made by the robot. If $x \geqslant n - a$ then the robot finds one endpoint before the first return and the total number of traversals is $2n - x = opt(L_n, v)$. If $x < n - a$ then the robot makes $2a + x$ traversals before reaching the first endpoint and additional n traversals to reach the second endpoint, which gives a total of $2a+x+n$ edge traversals. In order to compute the overhead, we need to compute $opt(L_n, v)$. If $x \leqslant n/2$ then $opt(L_n, v) = n+x$ and the ratio $\frac{\mathcal{C}(\texttt{Unanchored-Line}, L_n, v)}{opt(L_n, v)}$ is maximized for $x = 1$ giving

$$\frac{\mathcal{C}(\texttt{Unanchored-Line}, L_n, v)}{opt(L_n, v)} \leqslant \frac{2\lfloor(\sqrt{3}-1)n/2\rfloor + 1 + n}{1+n} \leqslant \frac{\sqrt{3}n+1}{n+1} \leqslant \sqrt{3}.$$

If $x > n/2$ then $opt(L_n, v) = 2n - x$ and the ratio $\frac{\mathcal{C}(\texttt{Unanchored-Line}, L_n, v)}{opt(L_n, v)}$ is maximized for $x = n - a - 1$ giving

$$\frac{\mathcal{C}(\texttt{Unanchored-Line}, L_n, v)}{opt(L_n, v)} \leqslant \frac{2a+x+n}{n+a+1} = \frac{\lfloor(\sqrt{3}-1)n/2\rfloor + 2n - 1}{n + \lfloor(\sqrt{3}-1)n/2\rfloor + 1}$$

$$\leqslant \frac{(\sqrt{3}-1)n/2+2n-1}{n+(\sqrt{3}-1)n/2} \leqslant \frac{(3+\sqrt{3})n/2}{(\sqrt{3}+1)n/2}$$
$$= \frac{3+\sqrt{3}}{\sqrt{3}+1} = \sqrt{3}.$$

Thus $\mathcal{O}_{\mathcal{L}}(\texttt{Unanchored-Line}) \leqslant \sqrt{3}$. □

Theorem 2.4. *For all algorithms $\mathcal{A}$ with an unanchored map, $\mathcal{O}_{\mathcal{L}}(\mathcal{A}) \geqslant \sqrt{3}$.*

Proof. The proof is divided in two parts. First we show that for every exploration algorithm of the line with an unanchored map, there is an algorithm that does at most two returns and has equal or smaller overhead. In the second part we show that the overhead is at least $\sqrt{3}$ for all algorithms with at most two returns.

For each value of n, all regular algorithms can be classified according to the maximum number of returns performed while exploring the line L_n before reaching an endpoint. Fix $n \geqslant 11$ and let *type k* be the set of algorithms that always do at most k returns before reaching an endpoint, and that do exactly this many returns for some combination of starting node and (adversary) choice of direction. Notice that one algorithm can be of different types for different values of n, and that algorithm Unanchored-Line is of type 1 for every $n \geqslant 3$. We now show that for every exploration algorithm $\mathcal{A}$ there exists an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ is of type 1 and $\max_{v \in L_n}(\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)}) \leqslant \max_{v \in L_n}(\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)})$.

Depth-first search is the only algorithm of type 0. Since $\mathcal{O}_{\{L_n\}}(\texttt{DFS}) \geqslant \frac{21}{12} \geqslant \sqrt{3}$, when $n \geqslant 11$, it follows that the algorithm Unanchored-Line is an algorithm of type 1 with the required property, for algorithms of type 0.

Consider algorithms of type 2. Every such algorithm $\mathcal{A}$ can be described as follows (assuming that no endpoint is encountered before the first two returns):

- Traverse a edges in one direction.
- Return and traverse $a+b$ edges in the opposite direction.
- Return again and go to the endpoint.
- Return and go to the other endpoint.

We show that $a=0$ minimizes the overhead for $\mathcal{A}$, which in effect proves that for every algorithm of type 2 there is an algorithm of type 1 with equal or smaller overhead. Let x be the distance from v to the endpoint in the direction of the first traversal. There are three ranges of values of x that are of interest for calculating the overhead of $\mathcal{A}$. When $x \leqslant a$, then $\mathcal{A}$ makes $x+n$ traversals, we call this case 1. When $a < x < n-b$, then the number of traversals is $2a+2b+x+n$, this is case 2. Finally, when $x \geqslant n-b$, then the number of traversals is $2a+2n-x$, this is case 3. Clearly $opt(L_n, v) = \min(n+x, 2n-x)$. Observe that for $a > \frac{n}{2}$, case 1 is dominated by case 3 (choose $x = n-1$), and since $\frac{x+n}{\min(n+x, 2n-x)} = 1$ when $a \leqslant \frac{n}{2}$, it follows that case 1 never dominates and can be discarded from our considerations. In case 3, $opt(L_n, v)$ does not depend on a and $a=0$ minimizes the ratio in this case. Case 2 is divided into two subcases. When $a < x \leqslant \frac{n}{2}$, we get $\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)} = \frac{2a+2b+x+n}{x+n}$ which is maximized for $x = a+1$, giving $\frac{3a+2b+n+1}{n+a+1}$. When $\frac{n}{2} < x < n-b$, we get $\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)} = \frac{2a+2b+x+n}{2n-x}$ which is maximized for $x = n-b-1$, giving $\frac{2a+b+3n-1}{n+b+1}$. In both cases, this maximum value of the ratio $\frac{\mathcal{C}(\mathcal{A},L_n,v)}{opt(L_n,v)}$ is minimized for $a = 0$. Hence for every algorithm of type 2 there is an algorithm of type 1 with equal or smaller overhead.

Now we consider algorithms of type $k > 2$. Let $\mathcal{A}$ be such an algorithm. Assuming that no endpoint is encountered before the first 3 returns, the initial behavior of $\mathcal{A}$ can be described as follows:

- Traverse a edges in one direction.
- Return and traverse $a + b$ edges in the opposite direction.
- Return and traverse $b + c$ edges in the first direction.
- Return.

Note that $c > a$ and $b+c \leqslant n-2$. Let $\mathcal{A}'$ be the algorithm of type $k-1$ that behaves exactly like $\mathcal{A}$ but for which $a = 0$. Let S be the set of nodes $v \in L_n$ such that v is at distance larger than b from one endpoint and at distance larger than c from the other.

Claim 1. *$\mathcal{C}(\mathcal{A}', L_n, v) \leqslant \mathcal{C}(\mathcal{A}, L_n, v)$, for any $v \in S$.*

Let $v \in S$ and suppose that A and B are endpoints such that $dist(v, A) > b$ and $dist(v, B) > c$. Let $z = dist(v, A)$. Consider two cases.

Case 1: $a \leqslant b$.

In this case $\mathcal{A}$ does not encounter an endpoint before the first return. Hence $\mathcal{A}'$ performs $2a$ fewer traversals than $\mathcal{A}$, regardless of the choice of the initial direction. Hence Claim 1 holds in this case.

Case 2: $a > b$.

In this case we also have $b < c$. If $\mathcal{A}'$ starts towards A then $\mathcal{A}$ starts towards B, which is at distance larger than a from v. Hence $\mathcal{A}$ does not encounter an endpoint before the first return. Consequently $\mathcal{A}'$ performs $2a$ fewer traversals than $\mathcal{A}$.

Suppose that $\mathcal{A}'$ starts towards B. If $z > a$ then $\mathcal{A}$ (starting towards A) does not encounter an endpoint before the first return. Consequently $\mathcal{A}'$ performs $2a$ fewer traversals than $\mathcal{A}$. If $z \leqslant a$ then also $z \leqslant c$. Hence $\mathcal{A}'$ (starting towards B) performs $2b + z + n \leqslant 2b + c + n$ traversals. So in this case starting towards A instead of B results in more traversals for $\mathcal{A}'$: at least $3b + 2c + n$. But, as we showed before, $\mathcal{A}$ performs $2a$ more traversals than $\mathcal{A}'$ for this initial direction.

Hence, for all cases and for any decision of the adversary concerning the initial direction of $\mathcal{A}'$, we showed that some decision of the adversary concerning the initial direction of $\mathcal{A}$ yields more traversals. This completes the proof of Claim 1.

Claim 2.

$$\max_{v \in S} \left(\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)} \right) \geqslant \frac{\mathcal{C}(\mathcal{A}', L_n, v')}{opt(L_n, v')}$$

for any $v' \notin S$.

Consider two cases.

Case 1: $b < c$.

Choose a starting node v at distance $b+1$ from an endpoint and choose the initial direction of algorithm $\mathcal{A}'$ towards this endpoint. Since $b + c \leqslant n-2$, we have $v \in S$. By the choice of the direction, the robot does not encounter an endpoint before returning twice. The number of traversals in algorithm $\mathcal{A}'$ for this choice of direction is at least $3b + 2c + n + 1$, hence we have $\mathcal{C}(\mathcal{A}', L_n, v) \geqslant 3b + 2c + n + 1$. On the other hand, $opt(L_n, v) = n + b + 1$, since the closest endpoint is at distance $b + 1$ from v. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)} \geqslant \frac{3b + 2c + n + 1}{n + b + 1}.$$

Now take a node $v' \notin S$. Let x be the distance from v' to the closest endpoint A and y the distance from v' to the other endpoint B. Hence either $x < b$ or $y < c$.

Case 1.1: $x < b$.

If $\mathcal{A}'$ starts towards A then it performs $x + n$ traversals. Otherwise it performs $2b + x + n$ traversals. Hence $\mathcal{C}(\mathcal{A}', L_n, v') \leqslant 2b + x + n$. On the other hand, $opt(L_n, v) = n + x$. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v')}{opt(L_n, v')} \leqslant \frac{2b + x + n}{n + x}.$$

We have

$$\frac{3b + 2c + n + 1}{n + b + 1} \geqslant \frac{5b + n + 2}{n + b + 1} \geqslant \frac{2b + n}{n} \geqslant \frac{2b + x + n}{n + x}.$$

This proves Claim 2 in this case.

Case 1.2: $y < c$.

In this case $x > b$. If $\mathcal{A}'$ starts towards A then it performs $2b + y + n$ traversals. Otherwise it performs at most $2b + x + n$ traversals. Hence $\mathcal{C}(\mathcal{A}', L_n, v') \leqslant 2b + y + n$. As before, $opt(L_n, v) = n + x$. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v')}{opt(L_n, v')} \leqslant \frac{2b + y + n}{n + x}.$$

We have $n + b + 1 \leqslant n + x$ and $3b + 2c + n + 1 \geqslant 2b + y + n$. Hence

$$\frac{3b + 2c + n + 1}{n + b + 1} \geqslant \frac{2b + y + n}{n + x}.$$

This proves Claim 2 in this case.

Case 2: $c \leqslant b$

Choose a starting node v at distance $c + 1$ from an endpoint and choose the initial direction of algorithm $\mathcal{A}'$ towards the other endpoint. As in Case 1, the number of traversals in algorithm $\mathcal{A}'$ for this choice of direction is at least $3b + 2c + n + 1$, hence we have $\mathcal{C}(\mathcal{A}', L_n, v) \geqslant 3b + 2c + n + 1$. On the other hand, $opt(L_n, v) = n + c + 1$, since the closest endpoint is at distance $c + 1$ from v. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)} \geqslant \frac{3b + 2c + n + 1}{n + c + 1}.$$

Now take a node $v' \notin S$. Let x be the distance from v' to the closest endpoint A and y the distance from v' to the other endpoint B. Hence either $x < c$ or $y < b$.

Case 2.1: $x < c$.

If $\mathcal{A}'$ starts towards A then it performs $x+n$ traversals. Otherwise it performs $2b+x+n$ traversals. Hence $\mathcal{C}(\mathcal{A}', L_n, v') \leqslant 2b + x + n$. On the other hand, $opt(L_n, v) = n + x$. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v')}{opt(L_n, v')} \leqslant \frac{2b+x+n}{n+x}.$$

We have

$$\frac{3b+2c+n+1}{n+c+1} = 1 + \frac{3b+c}{n+c+1} \geqslant 1 + \frac{2b}{n} \geqslant \frac{2b+x+n}{n+x}.$$

This proves Claim 2 in this case.

Case 2.2: $y < b$.

In this case $x > c$. If $\mathcal{A}'$ starts towards A then it performs $x+n$ traversals. Otherwise it performs $y+n$ traversals. Hence $\mathcal{C}(\mathcal{A}', L_n, v') \leqslant y + n$. As before, $opt(L_n, v) = n + x$. Hence

$$\frac{\mathcal{C}(\mathcal{A}', L_n, v')}{opt(L_n, v')} \leqslant \frac{y+n}{n+x}.$$

We have $n+c+1 \leqslant n+x$ and $3b+2c+n+1 \geqslant y+n$, hence

$$\frac{3b+2c+n+1}{n+c+1} \geqslant \frac{y+n}{n+x}.$$

This proves Claim 2 in this case and completes its proof.

By Claim 1 we have

$$\max_{v\in S}\left(\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)}\right) \leqslant \max_{v\in S}\left(\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)}\right) \leqslant \mathcal{O}_{L_n}(\mathcal{A}).$$

By Claim 2 we have

$$\max_{v\in S}\left(\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)}\right) = \mathcal{O}_{L_n}(\mathcal{A}').$$

Hence $\mathcal{O}_{L_n}(\mathcal{A}') \leqslant \mathcal{O}_{L_n}(\mathcal{A})$.

Thus, for any algorithm $\mathcal{A}$ of type k, we have shown an algorithm $\mathcal{A}'$ of type $k-1$ such that the overhead of $\mathcal{A}'$ for the line L_n is equal to or less than the overhead of $\mathcal{A}$.

It follows by induction that for any regular algorithm $\mathcal{A}$ there exists an algorithm $\mathcal{A}'$ of type 1 such that $\max_{v\in L_n}(\frac{\mathcal{C}(\mathcal{A}', L_n, v)}{opt(L_n, v)}) \leqslant \max_{v\in L_n}(\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)})$, which concludes the first part of the proof.

Now it is enough to prove that for any algorithm $\mathcal{A}'$ of type 1 that performs a edge traversals before the first return, there exists a starting node v such that $\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)} \geqslant \sqrt{3} - \frac{c}{n}$ for some positive constant c. First assume that $a \leqslant \frac{\sqrt{3}-1}{2}n$. Choose the starting node v at distance $a+1$ from an endpoint p of the line. The adversary chooses the direction of p for the first traversal. The algorithm performs $a+2n-1$ traversals and $opt(L_n, v) = a+n+1$, giving $\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)} = \frac{a+2n-1}{a+n+1} \geqslant \frac{a+2n+\sqrt{3}}{a+n+1} - \frac{c}{n} \geqslant \sqrt{3} - \frac{c}{n}$, for $a \leqslant \frac{\sqrt{3}-1}{2}n$ and a sufficiently large

n. If, on the other hand, $a > \frac{\sqrt{3}-1}{2}n$, choose v at distance 1 from the closest endpoint p. The adversary chooses the opposite direction of p for the first traversal. The algorithm performs $2a+1+n$ traversals and $opt(L_n, v) = 1+n$, giving $\frac{\mathcal{C}(\mathcal{A}, L_n, v)}{opt(L_n, v)} = \frac{2a+1+n}{1+n} \geqslant \frac{1+\sqrt{3}n}{1+n} \geqslant \sqrt{3} - \frac{c}{n}$ for $a > \frac{\sqrt{3}-1}{2}n$ and a sufficiently large n. □

2.3. Exploration without a map

We finally consider the scenario when no map is available to the robot, i.e., the robot has no information about the line whatsoever: it knows neither its length, nor the position of the starting node. We prove that in this case the overhead of every exploration algorithm for the class $\mathcal{L}$ of lines is at least 2, and consequently depth-first search with overhead 2 is optimal.

Theorem 2.5. *Every exploration algorithm without a map has overhead at least* 2 *for the class* $\mathcal{L}$ *of lines.*

Proof. Consider any (regular) exploration algorithm $\mathcal{E}$. Call the first direction chosen by the adversary the right direction. By regularity of $\mathcal{E}$ there are three possible cases for the initial part of the run of $\mathcal{E}$ before an endpoint is reached for the first time, corresponding to three types of algorithms:

Type 1. There exist two infinite strictly increasing sequences $(a_1, a_2, \ldots)$ and $(b_1, b_2, \ldots)$ of natural numbers, such that the robot goes a_1 steps right from the starting node, then goes back to it, then goes b_1 steps left from the starting node, then goes back to it, then goes a_2 steps right from the starting node, then goes back to it, then goes b_2 steps left from the starting node, etc., until an endpoint is reached for the first time.

Type 2. There exist two strictly increasing sequences $(a_1, a_2, \ldots, a_i)$ and $(b_1, b_2, \ldots, b_{i-1})$ of natural numbers, such that the robot goes a_1 steps right from the starting node, then goes back to it, then goes b_1 steps left from the starting node, then goes back to it, then goes a_2 steps right from the starting node, then goes back to it, then goes b_2 steps left from the starting node, etc., then goes a_i steps right from the starting node, and then goes left till the endpoint.

Type 3. There exist two strictly increasing sequences $(a_1, a_2, \ldots, a_i)$ and $(b_1, b_2, \ldots, b_i)$ of natural numbers, such that the robot goes a_1 steps right from the starting node, then goes back to it, then goes b_1 steps left from the starting node, then goes back to it, then goes a_2 steps right from the starting node, then goes back to it, then goes b_2 steps left from the starting node, etc., then goes a_i steps right from the starting node, then goes back to it, then goes b_i steps left from the starting node, and then goes right till the endpoint.

We will show that each of the above three types of exploration algorithms has overhead at least 2.

Algorithms of type 1. By regularity we have $b_1 \geqslant 1$. Let $a = a_2$, $b = b_1$, and let the line L_n be $[-b-1, -b, \ldots, 0, \ldots, a, a+1]$, where 0 is the starting node and positive numbers go in the right direction. The line has length $n = a+b+2$. We have $\mathcal{C}(\mathcal{E}, L_n, 0) \geqslant 3a+4b+5$. This is proved as follows. By the time the robot makes the second turn left it already made at least $2+2b+a$ steps. Then it makes at least $a+b$ additional steps left, by regularity.

At this point it is at distance 1 from the left endpoint, and the right endpoint is yet unexplored. Hence at least $n + 1 = a + b + 3$ additional steps are needed, for a total of $3a + 4b + 5$ steps.

On the other hand, $opt(L_n, 0) = 2b + a + 3$, if $b \leqslant a$ and $opt(L_n, 0) = 2a + b + 3$, if $a \leqslant b$. In the first case we have

$$\frac{\mathcal{C}(\mathcal{E}, L_n, 0)}{opt(L_n, 0)} \geqslant 2$$

in view of $a \geqslant 1$, and in the second case we have

$$\frac{\mathcal{C}(\mathcal{E}, L_n, 0)}{opt(L_n, 0)} \geqslant \frac{4a + 2b + 6}{2a + b + 3} \geqslant 2$$

in view of $b \geqslant 1$ and of $a \leqslant b$. This proves $\mathcal{O}_{\mathcal{L}}(\mathcal{E}) \geqslant 2$ for algorithms of type 1.

Algorithms of type 2. It is enough to show that, for any $\varepsilon > 0$, there exists a line L_n, and a position of the starting node v in it, such that

$$\frac{\mathcal{C}(\mathcal{E}, L_n, v)}{opt(L_n, v)} \geqslant 2 - \varepsilon.$$

Fix an $\varepsilon > 0$, and the index i given by the algorithm (the index of the last turn left before going indefinitely left, until the endpoint is reached). Let $a = a_i$. Let n be a positive integer larger than $a+1$, such that $(2n+a-1)/(n+a+1) \geqslant 2-\varepsilon$. (Such an integer n exists, since, for any fixed a, this fraction converges to 2 as n grows.) Let $k = n - a - 1$. Let the line L_n be $[-k, -k + 1, \ldots, 0, \ldots, a, a + 1]$, where 0 is the starting node and positive numbers go in the right direction. We have $\mathcal{C}(\mathcal{E}, L_n, 0) \geqslant 2n + a - 1$. Indeed, by the last turn left before going indefinitely left, the robot makes at least a steps. Then it makes $n - 1$ steps to reach the left endpoint, and still has to make n more steps to reach the right endpoint, yet unexplored.

On the other hand, $opt(L_n, 0) \leqslant n + a + 1$. Hence we have

$$\frac{\mathcal{C}(\mathcal{E}, L_n, 0)}{opt(L_n, 0)} \geqslant \frac{2n + a - 1}{n + a + 1} \geqslant 2 - \varepsilon.$$

This proves $\mathcal{O}_{\mathcal{L}}(\mathcal{E}) \geqslant 2$ for algorithms of type 2. For algorithms of type 3 the proof is analogous to that for type 2. Thus we have shown that $\mathcal{O}_{\mathcal{L}}(\mathcal{E}) \geqslant 2$ for all exploration algorithms. $\square$

3. Arbitrary trees

In this section we describe an optimal algorithm for tree exploration with an anchored map and prove that its overhead is $\frac{3}{2}$ for the class $\mathcal{T}$ of all trees. As for the scenario with an unanchored map, we show that the optimal algorithm has overhead strictly smaller than 2, and hence it is not DFS. (By Theorem 2.4, the optimal overhead for the class $\mathcal{T}$ cannot be smaller than $\sqrt{3}$.) Notice that if no map is available then depth-first search is an

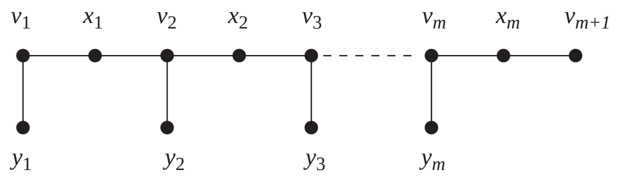

Fig. 1. The tree T.

optimal algorithm for exploring the class of all trees. Its overhead is 2. This follows from Theorem 2.5.

3.1. *Exploration with an anchored map*

Lemma 3.1. *For all algorithms $\mathcal{A}$ with an anchored map, $\mathcal{O}_{\mathcal{T}}(\mathcal{A}) \geqslant 3/2$.*

Proof. We construct a tree T of arbitrarily large size n, with starting node v_1, for which $\frac{\mathcal{C}(\mathcal{A},T,v_1)}{opt(T,v_1)} \geqslant \frac{3}{2} - \frac{c}{n}$, for some constant c and any algorithm $\mathcal{A}$. The tree T of size $n = 3m+1$ is defined by the set of nodes $V(T) = \{v_1, \dots, v_{m+1}, x_1, \dots, x_m, y_1, \dots, y_m\}$ and the set of edges $E(T) = \{[v_i, y_i], [v_i, x_i], [x_i, v_{i+1}] : 1 \leqslant i \leqslant m\}$ (see Fig. 1).

Clearly $opt(T, v_1) = 4m$: every edge $[v_i, y_i]$ is traversed twice and the remaining edges only once. Consider a robot that has an anchored map. When the robot is in v_i and the edges $[v_i, y_i]$ and $[v_i, x_i]$ are both unexplored, the adversary chooses the edge $[v_i, x_i]$ when $\mathcal{A}$ decides to use an unexplored edge. Let $i_0 \leqslant m$ be the integer such that the robot concludes exploration in y_{i_0} or v_{i_0+1}. For all $i \in \{1, \dots, m\} \backslash \{i_0\}$, the robot traverses the edges $[v_i, y_i]$, $[v_i, x_i]$ and $[x_i, v_{i+1}]$ no less than 6 times in total: the edge $[v_i, y_i]$ twice and either $[v_i, x_i]$ 3 times and $[x_i, v_{i+1}]$ once (if it returns immediately), or each of the two edges $[v_i, x_i]$ and $[x_i, v_{i+1}]$ twice (otherwise). For i_0, the number is at least 5, giving the ratio $\frac{\mathcal{C}(\mathcal{A},T,v_1)}{opt(T,v_1)} \geqslant \frac{6m-1}{4m}$, which proves the lemma. □

We now present an optimal algorithm for exploring a tree T with an anchored map. Let D be the eccentricity of the starting node v. Consider all elementary paths of length D starting at v. Two such paths $P_1 = (v_0 = v, v_1, \dots, v_D)$ and $P_2 = (v'_0 = v, v'_1, \dots, v'_D)$ are called isomorphic if there exists an automorphism of T such that $f(v_i) = v'_i$, for all $i = 0, \dots, D$.

Algorithm Anchored-Tree

Choose one node on the map of T at distance D from v. Let P be the path on the map from v to this node. Perform a depth-first search with the following adjustments. Suppose that at some point of the exploration the robot, using the map, can determine that its current position corresponds to a node u on a path isomorphic to P, and there is at least one visited node different from u, with unexplored edges. Call this situation a *break*. When a break occurs, continue depth-first search in the subtree T' containing u and resulting from removal of all unexplored edges incident to u. Call this procedure a *limited* depth-first search. When no unexplored edges remain in this subtree, resume "standard" depth-first search, i.e., move to u and continue depth-first search in the rest of the tree, until the next break. (Notice that many breaks can occur during the exploration.) The robot stops when there are no more unexplored edges.

Lemma 3.2. *Algorithm Anchored-Tree explores any tree T of size n with starting node v, using at most $4(n-1)-3D$ edge traversals.*

Proof. It is clear that the robot traverses every edge outside of P twice. The edges on P are traversed at least once. After every break, when the robot interrupts depth-first search and performs a limited depth-first search, some edges on P are traversed 2 more times. After completing a limited depth-first search in a subtree T', started at u, the tree T' is entirely explored, thus no edges on P are traversed more than 3 times. It remains to count the number of edges on P that are traversed 3 times. An edge e on P is traversed during standard depth-first search only if there are either no other unexplored edges incident to visited nodes, or if the robot cannot determine that it is on a path isomorphic to P (otherwise a break occurs). In the first case, e will not be traversed again hence the total number of its traversals is 1. For the second case to occur, there must be at least one more edge on the map at the same distance from v, or else the robot could determine that it is on a path isomorphic to P. Only such edges on P can be traversed 3 times. Thus the number of edges on P traversed 3 times is bounded by the number of edges not on P, i.e., by $n-1-D$. Consequently we have three groups of traversals.

(1) $n-1-D$ edges outside of P are traversed exactly twice, contributing $2(n-1-D)$ traversals.
(2) First traversals of edges on P, a total of D traversals.
(3) Two additional traversals of at most $n-1-D$ edges on P, a total of $2(n-1-D)$ traversals.

Thus, the total number of traversals is at most $4(n-1)-3D$. □

We use a modified version of Anchored-Tree that runs Anchored-Tree if $D > 2n/3$, and otherwise runs DFS.

Theorem 3.1. $\mathcal{O}_{\mathcal{T}}(\texttt{Modified-Anchored-Tree}) = 3/2$.

Proof. If $D \leqslant 2n/3$, the ratio is $\frac{\mathcal{C}(\texttt{DFS},T,v)}{opt(T,v)} \leqslant \frac{2(n-1)-1}{2(n-1)-2n/3} \leqslant 3/2$. If $D > 2n/3$, the ratio is $\frac{\mathcal{C}(\texttt{Anchored-Tree},T,v)}{opt(T,v)} \leqslant \frac{4(n-1)-3D}{2(n-1)-D}$ which is maximized for $D = 2n/3$, giving the ratio $\frac{4(n-1)-2n}{2(n-1)-2n/3} < 3/2$. □

3.2. Exploration with an unanchored map

We now show that an optimal algorithm with an unanchored map has overhead strictly smaller than 2, and thus it is not DFS. We do not make any attempt at optimizing the constant, and show an algorithm with overhead at most 1.99. We first present an algorithm that improves on depth-first search for trees of high diameter and at least 100 nodes.

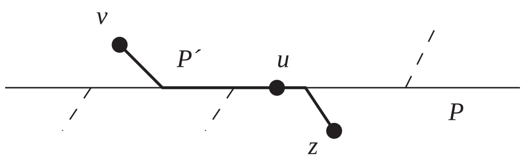

Fig. 2. Nodes visited in phase 1 of Algorithm Unanchored-Tree.

Algorithm Unanchored-Tree

The algorithm explores a tree T of size $n \geqslant 100$ and diameter $D \geqslant 0.99n$. It works in three phases. Let v be the starting node and let $a = \lfloor 0.3n \rfloor$. Let P be a path of length D on the map.

(1) Perform depth-first search until a node z is found at distance a from v. Let P' be the path from v to z, and let u be the node on P' at distance $\lceil a - 0.01n \rceil$ from v (see Fig. 2). Move to u.

(2) Perform a partial depth-first search to explore all unvisited nodes in the subtree T' of T containing v and resulting from the removal of u. Then return to u.

(3) Perform depth-first search in the remaining part of tree T, with the following modification. If the robot, using the map, can identify a nonempty set S of nodes on P such that the current position w corresponds to an element of this set (i.e. the robot "knows" that it is on P), and there is at least one visited node different from w with incident unexplored edges, we say that a *break* occurs. When a break occurs, continue depth-first search in the subtree containing u and resulting from removal of all unexplored edges incident to w. Call this a *limited* depth-first search. When no unexplored edges remain in this subtree, return to w and resume depth-first search in the remaining part of the tree, until the next break. (Notice that many breaks can occur during the exploration.) The robot stops when there are no more unexplored edges.

Lemma 3.3. *Let $\mathcal{T}^*$ be the class of trees with $n \geqslant 100$ and $D \geqslant 0.99n$. We have $\mathcal{O}_{\mathcal{T}^*}(\texttt{Unanchored-Tree}) \leqslant 1.99$.*

Proof. Let x be the number of nodes in the subtree T' (defined in phase 2 of the algorithm). We first compute the total number of edge traversals performed by the algorithm, by looking at each phase separately.

(1) Since $D \geqslant 0.99n$ and $a = \lfloor 0.3n \rfloor$, clearly some edges of P are traversed during phase 1 and $P \cap P'$ will contain at least $\lfloor 0.3n \rfloor - \lfloor 0.01n \rfloor \geqslant \lfloor 0.29n \rfloor$ nodes. The node u at distance $\lceil a - 0.01n \rceil$ from an endpoint of P' must therefore be on P. Assume that during this phase no endpoint of P is visited. (The other case will be discussed separately.) As there are no more than $\lfloor 0.01n \rfloor$ edges outside of P, a node at distance a is reached using at most $0.02n + a$ edge traversals. Going back to u requires additional $\lfloor 0.01n \rfloor$ traversals, for a total of $0.03n + a$ traversals.

(2) This phase is a straightforward depth-first search of a tree with $x - 1$ edges. Counting the return to u, the number of traversals required is at most $2x$.

(3) During the last phase, in the subtree $T \backslash T'$, edges outside of P are traversed exactly twice and edges on P are traversed at least once. No edges on P are traversed more than 3 times for the following reason. During "standard" depth-first search, every edge on P is traversed exactly once. During a single limited depth-first search, every edge on P is traversed at most twice. Since the limited depth-first search explores the entire subtree resulting from a break in depth-first search, no edges on P are traversed in subsequent calls of limited depth-first search. Thus every edge on P is traversed at most 3 times (in this phase). It remains to count the number of edges on P that are traversed 3 times; call these edges *special*. In order to do this, we need to study under what circumstances a break occurs. Since the node u is on P, one endpoint of P has been visited during depth-first search performed in phase 2. The distance to this endpoint is known to the robot during phase 3. At any point of the exploration, there are no more than 2 possible nodes on P (on the map) that can correspond to the current position of the robot. (In Fig. 3, A and B are such nodes.) A break can occur only when the robot is able to exclude all other nodes on the map as possible current locations. (In the situation depicted in Fig. 3, node C cannot be excluded, and thus prevents a break.) Suppose that at some point of the exploration, there exist visited nodes, other than the current location, with unexplored incident edges. To every edge e outside of P we can assign at most two special edges, e' and e'', such that the existence of e on the map causes e' and e'' to be special. If the robot is at node A (see Fig. 3), it traverses e' during standard depth-first search because it cannot distinguish e' from e. At a later point of standard depth-first search the robot traverses e'' for the same reason. (Both e' and e'' are then traversed during limited depth-first search twice each.)
The total number of edges on P that are traversed 3 times during this phase is thus bounded by $0.02n$. There are $n-1-x$ edges in the subtree explored during this phase. The total number of traversals during phase 3 can be estimated as follows: at most $0.02n$ traversals of edges outside of P, at most $n-1-x$ first traversals of edges on P and finally at most $0.04n$ extra traversals of special edges (2 extra traversals of each of at most $0.02n$ edges), for a total of $1.06n-1-x$ traversals.

The total number of traversals for all three phases is not more than $1.09n+a+x$. In the case when an endpoint of P is visited during phase 1, observe that while some edges on P are traversed 2 times during phase 1 in addition to the above stated number of traversals, exactly the same number of traversals are saved in phase 2. Thus, the estimate $1.09n+a+x$ on the number of traversals holds also in this case (phase 3 remains unaffected).

We now need to calculate $opt(T,v)$ which depends on x. The value of $opt(T,v)$ is $2(n-1)-ecc(v)$. We have $ecc(v) \leqslant \max(x-a+0.02n, n-x+a-0.01n)$. This gives $\mathcal{O}_{\mathcal{T}}(\texttt{Unanchored-Tree}) \leqslant \frac{1.39n+x}{\min(2.28n-3-x, 0.71n-2+x)}$ which is strictly less than 1.99 for $n \geqslant 100$. □

It is easy to verify that $\frac{\mathcal{C}(\texttt{DFS},T,v)}{opt(T,v)} \leqslant 1.99$ for all trees with less than 100 nodes and for all trees of diameter $D < 0.99n$. Together with Lemma 3.3 this gives the following theorem.

Theorem 3.2. *There exists an algorithm $\mathcal{A}$ with an unanchored map, for which $\mathcal{O}_{\mathcal{T}}(\mathcal{A}) \leqslant 1.99$.*

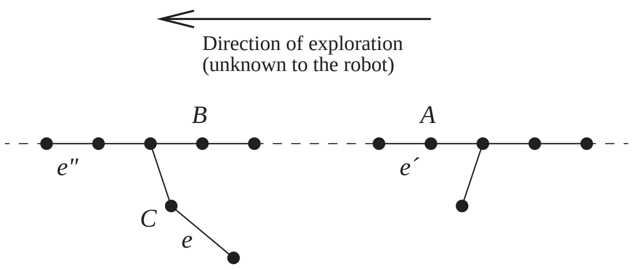

Fig. 3. Edge e causes edges e' and e'' to be traversed 3 times.

Recall that Theorem 2.4 implies $\mathcal{O}_{\mathcal{T}}(\mathcal{A}) \geqslant \sqrt{3}$, for all algorithms with an unanchored map. The construction of an optimal exploration algorithm with an unanchored map, for the class of trees, and establishing the value of the best overhead remains an open problem.

4. Arbitrary graphs

In this section we consider the class $\mathcal{G}$ of arbitrary undirected connected graphs, and prove that the overhead of any exploration algorithm for this class is at least 2, under all three scenarios. This implies that, for the class $\mathcal{G}$, depth-first search is optimal. Since the scenario with an anchored map provides most information among all three scenarios, it is enough to prove this lower bound under this scenario. To this end, we construct a class of Eulerian graphs S_m of arbitrarily large size, each with a distinguished starting node x, such that for any exploration algorithm $\mathcal{A}$, $\mathcal{C}(\mathcal{A}, S_m, x) \geqslant 2e - o(e)$, where e is the number of edges in S_m. Since $opt(S_m, x) = e$, this will prove our result.

The building blocks of graphs S_m are graphs called *thick lines*, defined in [17]. A thick line L of length n is a graph defined by the set of nodes $V(L) = \{v_0, v_1, \ldots, v_n, x_1, \ldots, x_n, y_1, \ldots, y_n\}$ and the set of edges $E(L) = \{[x_i, v_{i-1}],\ [x_i, v_i],\ [y_i, v_{i-1}],\ [y_i, v_i] : 1 \leqslant i \leqslant n\}$. The nodes v_0 and v_n are called the *ends* of L. For $i \in \{0, \ldots, n-1\}$, the cycle $[v_i, x_{i+1}, v_{i+1}, y_{i+1}, v_i]$ is called the *cycle connecting* v_i *and* v_{i+1}. We denote L by $v_0 \diamond v_1 \diamond \cdots \diamond v_n$. Notice that a thick line of length n is an Eulerian graph with $4n$ edges.

Thick lines were used in [17] to prove that DFS is optimal under the scenario with an unanchored map. (Notice that since a thick line L is an Eulerian graph, $opt(L, v)$ does not depend on v, and hence, in this special case, the measure from [17] coincides with our measure of overhead.) In fact, the following lemma is proved in [17].

Lemma 4.1. *Suppose that the robot starts at node v_0 of a thick line of length n and consider any exploration algorithm. Then there exists an adversary such that, when the robot reaches v_k, $k \in \{2, \ldots, n\}$, for the first time, then at least 6 moves have already been performed along the edges of the cycle connecting v_{k-2} and v_{k-1}.*

Lemma 4.1 was used in [17] to show that, under the scenario with an unanchored map, the cost of every exploration algorithm in a thick line of length n is at least $8n - 12$ which is enough to prove that overhead is at least 2. However, for the scenario with an anchored map, this is not the case. If the distances of the starting node from both ends of the thick line are known, there is a simple exploration algorithm with overhead 15/8. (It is based on the same idea as the algorithm for ordinary lines constructed in Section 2.) Thus, in order to strengthen the result from [17] to the scenario with an anchored map, we need the following, slightly more complicated class of graphs.

A *thick star* of radius m is a graph S_m consisting of m thick lines of length m, which have exactly one common node: one of the ends of each of these lines. Call this node x and consider it to be the starting node of the robot. All thick lines of length m attached to x are called *branches* of S_m.

Lemma 4.2. *For any exploration algorithm $\mathcal{A}$ with an anchored map, $\mathcal{C}(\mathcal{A}, S_m, x) \geqslant 8m^2 - \mathrm{o}(m^2)$.*

Proof. By Lemma 4.1, at the time when the robot reaches the other end of any branch it must use at least $6(m-1)+2$ edge traversals in this branch. At least $2m$ additional traversals are needed to return to the starting node x. This must be repeated at least $m-1$ times (there is no need of returning from the last branch), for a total of $(8m-4)(m-1) = 8m^2 - \mathrm{o}(m^2)$ edge traversals. □

Since every graph S_m is an Eulerian graph with $4m^2$ edges, this proves the following result.

Theorem 4.1. *For any exploration algorithm $\mathcal{A}$, $\mathcal{O}_{\mathcal{U}}(\mathcal{A}) = 2$, for the class $\mathcal{G}$ of all undirected connected graphs.*

Hence depth-first search is an optimal exploration algorithm for the class $\mathcal{G}$, under all three scenarios.

References

[1] S. Albers, M.R. Henzinger, Exploring unknown environments, SIAM J. Comput. 29 (2000) 1164–1188.
[2] B. Awerbuch, M. Betke, R. Rivest, M. Singh, Piecemeal graph learning by a mobile robot, in: Proc. 8th Conf. on Computer Learning Theory,1995, pp. 321–328.
[3] E. Bar-Eli, P. Berman, A. Fiat, R. Yan, On-line navigation in a room, J. Algorithms 17 (1994) 319–341.
[4] M.A. Bender, A. Fernandez, D. Ron, A. Sahai, S. Vadhan, The power of a pebble: exploring and mapping directed graphs, in: Proc. 30th Ann. Symp. on Theory of Computing,1998, pp. 269–278.
[5] M.A. Bender, D. Slonim, The power of team exploration: two robots can learn unlabeled directed graphs, in: Proc. 35th Ann. Symp. on Foundations of Computer Science,1994, pp. 75–85.
[7] M. Betke, R. Rivest, M. Singh, Piecemeal learning of an unknown environment, Mach. Learn. 18 (1995) 231–254.
[8] A. Blum, P. Raghavan, B. Schieber, Navigating in unfamiliar geometric terrain, SIAM J. Comput. 26 (1997) 110–137.
[9] X. Deng, T. Kameda, C.H. Papadimitriou, How to learn an unknown environment I: the rectilinear case, J. ACM 45 (1998) 215–245.
[10] X. Deng, A. Mirzaian, Competitive robot mapping with homogeneous markers, IEEE Trans. Robotics Automat. 12 (1996) 532–542.
[11] X. Deng, C.H. Papadimitriou, Exploring an unknown graph, J. Graph Theory 32 (1999) 265–297.

[12] K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree exploration with little memory, in: Proc. 13th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'2002), San Francisco, U.S.A.,January 2002, pp. 588–597.
[13] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, Robotic exploration as graph construction, IEEE Trans. Robotics Automat. 7 (1991) 859–865.
[14] V. Dujmović, S. Whitesides, On validating planar worlds, in: Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'2001), Washington DC, U.S.A,January 2001, pp. 791–792.
[15] C.A. Duncan, S.G. Kobourov, V.S.A. Kumar, Optimal constrained graph exploration, in: Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms,2001, pp. 807–814.
[16] P. Panaite, A. Pelc, Exploring unknown undirected graphs, J. Algorithms 33 (1999) 281–295.
[17] P. Panaite, A. Pelc, Impact of topographic information on graph exploration efficiency, Networks 36 (2000) 96–103.
[19] N.S.V. Rao, S. Hareti, W. Shi, S.S. Iyengar, Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, July 1993.