Smart Robot Teams Exploring Sparse Trees[⋆]

M. Dynia[1], J. Kutyłowski[2], F. Meyer auf der Heide[3], and C. Schindelhauer[4]

[1] DFG Graduate College "Automatic Configuration in Open Systems", Heinz Nixdorf Institute, University of Paderborn, Germany, mdynia@uni-paderborn.de

[2] International Graduate School of Dynamic Intelligent Systems, Heinz Nixdorf Institute, University of Paderborn, Germany, jarekk@uni-paderborn.de

[3] Heinz Nixdorf Institute, University of Paderborn, Germany, fmadh@uni-paderborn.de

[4] Computer Networks and Telematics, University of Freiburg, Germany, schindel@informatik.uni-freiburg.de

Abstract. We consider a tree which has to be completely explored by a group of k robots, initially placed at the root. The robots are mobile and can communicate using radio devices, but the communication range is bounded. They decide based on local, partial knowledge, and exchange information gathered during the exploration. There is no central authority which knows the graph and could control the movements of the robots – they have to organize themselves and jointly explore the tree.
The problem is that at every point of time the remaining unknown part of the tree may appear to be the worst case setting for the current deployment of robots. We present a deterministic distributed algorithm to explore T and we use a parameter of a tree called *density*. We compare the performance of our algorithm with the optimal algorithm having a-priori knowledge of the same tree. We show that the above ratio is influenced only by the density and the height of the tree. Since the competitive ratio does not depend on the number of robots, our algorithm truly emphasizes the phenomena of self-organization. The more robots are provided, the faster the exploration of the terrain is completed.

1 Introduction

We study the problem of exploration of trees by a group of k mobile robots equipped with radio communication devices. The robots can move and can also communicate using radio devices, but the communication range is bounded. This allows a robot only a local view of the situation. Hence it has to make all of its decisions basing only on partial knowledge. One robot cannot explore the graph fast, so the group exchanges information gathered during the exploration. There

[⋆] This research is partially supported by the DFG-Sonderforschungsbereich SPP 1183: "Organic Computing. Smart Teams: Local, Distributed Strategies for Self-Organizing Robotic Exploration Teams" and by the EU within the 6th Framework Programme under contract 001907 (DELIS) is no central authority which knows the graph and could control the robots, so the team has to organize itself and jointly explore the tree, i.e. visit all its nodes. The problem is that at every point of time the remaining unknown part of the tree may appear to be the worst case setting for the current deployment of robots.

We use competitive analysis [1] to measure the performance of the algorithm. Let $C_{\mathcal{A}}(T)$ denote the running time of the online distributed algorithm $\mathcal{A}$ exploring a tree T not known in advance. Only outgoing edges are visible to a robot placed in some node, and it has to traverse an edge in order to recognize new parts of the graph. Since at each point of time the graph is known only partially, the adversary may construct the worst possible remaining part of the graph in an online fashion. In this model, traversing an edge takes one time step and costs of all other operations are neglected.

Let $C_{\mathrm{OPT}}(T)$ be the running time of the optimal algorithm which knows the exact structure of T and explores it optimally. The algorithm $\mathcal{A}$ is *σ-competitive* if for all T

$$C_{\mathcal{A}}(T) \leq \sigma \cdot C_{\mathrm{OPT}}(T) + \alpha \tag{1}$$

for some constant α.

Since robots use wireless devices with a bounded communication radius $\mathcal{A}$ is allowed to use only local communication, i.e. robot can communicate (once per a time step) only with those robots which are at distance one or less in the tree.

We can observe that the optimal offline algorithm does not have to use communication at all, since the graph is given in advance.

1.1 Related Work

The problem of exploring an unknown environment has been widely studied (see [2] for a survey) and usually the environment is modeled as a graph. Famous Traveling Salesman Problem and Chinese Postman Problem expose main problems of graph exploration.

The k-Traveling Salesman Problem (k-TSP) and k-Chinese Postman Problem (k-CPP) [3] are interesting extensions of these well known NP-hard problems[5]. All nodes (and edges) of a graph have to be visited (at least once) by one of the k robots initially placed at some node of the graph, subject to minimize the maximum length route of a robot. The contribution [5] captures the hardness of the k-TSP and shows how to construct (in polynomial time in size of a graph but exponential in k) optimal routes for an arbitrary k. In the location-allocation version of this problem an initial positions of robots have to be found as well.

There are several approximation algorithms of minmax k-TSP on a tree presented in [6, 4, 3, 7, 8, 5], but we should mention here that finding k-approximation is easy, and yields by taking a trivial optimal solution of 1-TSP (DFS traversal of a tree). Frederickson, Hecht and Kim [3] gave first worst-case analysis of k-TSP and developed $2 - 1/k$ approximation algorithm.

[5] also often called p-TSP and p-CPP, e.g. in [4]

In this work we consider an *online* version of k-TSP where a labeled graph is not known in advance, and the goal is to minimize overall time of an exploration. Authors of [9] and [10] study even harder problem and assume an unlabeled graph. If this setting one robot cannot explore the graph alone. Hence, in [10] the robot puts *pebbles* on the nodes of the graph in order to recognize visited ones, and in [9] it cooperates with other robot by exploration.

There are many publication considering an online exploration by a *single* robot (e.g. [11–14]), but the problem which exposes a real flavor is considering an exploring group of k robots [15, 16, 5].

Profits from a collective online exploration of *trees* are investigated in [15]. The authors prove competitive ratio of $\mathcal{O}(k/\log k)$ for the time of exploration compared to the time of an optimal algorithm which knows the tree. They also prove a lower bound of $2 - 1/k$ for this ratio. Moreover they study the influence of communication on the complexity of collective exploration. They prove that when no communication allowed, the competitive ratio is at least $\Omega(k)$. This shows that there is no cooperation at all, if no communication granted – the team explores in the same time that one robot would need to do it trivially.

Authors of [5] develop from their 2-approximation algorithm (similar to this in [3]) an *online* algorithm, subject to minimize *energy* used by a robot, rather than *time* of an exploration. They prove it uses at most 8 times the energy per a robot the optimal strategy for k robots does. They prove a lower bound of 1.5 for this energy model, which differs from the time model investigated in our work and in [15].

1.2 Our Results

We investigate a model with the cost to be an overall time of the collective tree exploration. Our results are two distributed, local algorithms, which explore an arbitrary tree using a group of k robots, and both exhibit a competitive ratio independent of k. It will solely depend on the tree, especially on its height and so called *density*.

Let us first define the *density* $p(T) \in \mathbb{N}$ for some tree T. It is the minimal natural number with the following property

$$\forall_{T'=(V',E')\subset T} \quad |V'| \leq 4 \cdot [h(T')]^{p(T)} \tag{2}$$

where $h(T)$ is the height of T. For many classes of trees an upper bound for $p(T)$ is known. For example trees embedded into a planar grid, fulfill $p(T) \leq 2$, and for binary trees $p(T) \leq h(T)/\log h(T)$. The density influences the running time of our algorithm. For simplicity of notation we will use D to denote the height $h(T)$ and p to denote $p(T)$.

The first distributed algorithm is presented in Sect. 2.2. It consists of two sub-algorithms, described and analyzed in the same section. It assumes only local communication and is fully online, i.e. no previous knowledge of the tree is required. It achieves a competitive ratio of

$$\mathcal{O}\left(D^{1-1/p} \cdot \min\left\{p, \log p \cdot D^{1/2p}\right\}\right)$$

where p is the density of the tree and D is its height.

The second algorithm described in Sect. 2.3 is also distributed and local. It has an improved competitive ratio of

$$\mathcal{O}\left(D^{1-1/p}\right) ,$$

but unfortunately it requires knowledge of the density value (or at least its constant approximation) before the exploration. It may be considered to be a drawback, but in fact, the value of density is known for many classes of trees.

For an arbitrary tree T we have $p(T) \leq \log n$ and thus our first algorithm is $\mathcal{O}(D \log\log n)$-competitive and the second algorithm is $\mathcal{O}(D)$-competitive. When we consider a class of trees which can be embedded in the s-dimensional grid it is easy to observe an upper bound $p(T) \leq 2s$. However, according to the definition (2) we have $p(T) \leq 2$ for a 2-dimensional grid and thus our algorithm is $\sqrt{D}$ competitive for this class of trees.

Unlike in [15], we present an algorithm with a competitive ratio which does not grow for large number of robots. It results in improved overall time of exploration for a team with increased number of robots. The coordination problem does not appear and thus our algorithm truly emphasizes the phenomena of robots cooperation. Additionally, our algorithms have an advantage over the one in [15], namely they are fully distributed and use only local communication. We prove that the exploration can also be efficient even under very bounded communication scheme.

2 The Online Algorithm

Only edges outgoing from the root are visible to a robot initially placed in the root of T, and it has to traverse an edge in order to recognize new parts of the graph. In our model, traversing an edge takes one time step and we neglect costs of all other operations. Robots communicate only if they meet in the same node – in this case in the root.

First, we present in Sect. 2.2 the *KarlsruheExpress-Clever* algorithm (*KE-Clever*) and then in Sect. 2.3 the *KE-Oblivious* algorithm. Both algorithms took their name from the place where the first steps of their design were made.

All presented algorithms are based on the same idea. Robots start in the root and explore the tree in so called chunks. They leave the root, enter the tree and after a current chunk is explored, they return to the root in order to exchange a valuable information and agree upon the strategy of exploration of the next chunk. Consecutive chunk lay further from the root than its predecessor – and in this way the progress of the exploration is realized. At some point of time the furthest leaf is reached by a robot and thus the tree is completely explored.

The team needs the information on the density of the tree in order to accurately estimate the amount of work a group can handle, i.e. the size of the chunk. The higher the density of the graph, the more job to do for robots, and thus the size of a chunk has to be reduced. On the other hand, small chunks weaken the progress of the exploration. The only aspect which distinguishes one algorithm from the other is the way they look for a good estimation of the destiny value. A proper balance between the size of the chunk and the efficiency of the exploration has to be found.

2.1 Definitions

The description of the algorithms as well as their performance analysis uses some notion which we define in this section (the circumference function ϕ, the *MyDFS* routine and finally the function r_{new}).

Given a rooted tree $T = (V, E)$ we denote by $h(T)$ the height of the tree, i.e. the number of nodes on a longest path from the root to a leaf. The *level* r of the tree is a set of nodes in distance r from the root. We describe nodes on each level, such that $v_r(i)$ denote the i-th node on level $r \in \mathbb{N}$ of T and $\phi(r)$ the number of nodes on that level. Let $T_v(h)$ be the subtree of T rooted at v and h in height, and let it be the subtree with maximal number of edges. Then, for some node $v \in V$ we define by *MyDFS*(v, r, p) the sequence of nodes created by the concatenation of the following sequences of nodes (Fig. 1) as

- the path from the root of T to the node v,
- the classical DFS algorithm for the tree $T_v(r^{1/p})$, broken after $8r$ steps (at node v'),
- the path from v' to v, and
- the path from v to the root of T.

We use *MyDFS*(v, r, p) only when the route from the root to the node v of length l is already known. In this case the execution takes $\mathcal{O}(r + l)$ time steps.

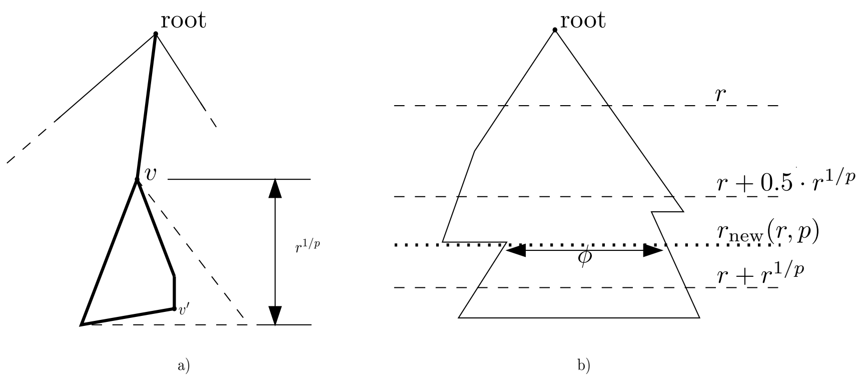

Fig. 1. Definition of a) MyDFS routine and b) the function r_{new}.

Moreover, we use the function r_{new} which dictates the progress of the presented algorithm. It is defined for a tree T and any number $r \geq 1$ and $p \geq 1$

$$r_{\text{new}}(r, p) = \operatorname{argmin} \left\{ \phi(j) : \quad j \in \left[r + \frac{1}{2} \cdot r^{1/p}, \ r + r^{1/p} \right] \right\} \tag{3}$$

Intuitively, the value of r_{new} points out the most narrow place in the tree within some specified interval (Fig. 1).

2.2 The *KE-Clever* Algorithm

First we present and analyze in detail the *KE-2* algorithm and then the *KE-1* algorithm which is a simple modification of the former. Both algorithms are the basic components of *KE-Clever* which we define and analyze at the end of this section.

The *KE-2* algorithm works in so called *epochs*. During one epoch all nodes in distance between some two well defined levels (*a chunk*) are being explored. A detailed description of the *KE-2* provides Algorithm 1 and Figure 2.

Algorithm 1 The *KE-2* algorithm

```
Require: $k$ robots placed in $ROOT$
 1: $r \leftarrow 1$
 2: $p' \leftarrow 1$
 3: repeat
 4:    $\phi \leftarrow \phi(r)$
 5:    for ($j \leftarrow 0$ to $\lceil \phi/k \rceil$) do
 6:       $v \leftarrow v_r(ID + k \cdot j \mod \phi)$
 7:       repeat
 8:          if (some RISE flag is set) then
 9:             $p' \leftarrow 2 \cdot p'$
10:          end if
11:          move and explore following $MyDFS(v, r, p')$ sequence
12:          if ($T_v(r^{1/p'})$ is not completely explored) then
13:             set RISE flag
14:          end if
15:       until (no flag RISE is set)
16:    end for
17:    $r \leftarrow r_{\text{new}}(r, p')$
18: until ($T$ is completely explored)
```

The algorithm maintains a variable r (the radius) whose value means that all levels of the tree up to the level r are already explored. It grows during the exploration, and the algorithm terminates when the tree is completely known. We call the lines 4–17 of the code an *epoch* of the algorithm. Epoch E_i starts with radius r_i and during the epoch all nodes on levels between r_i and r_{i+1} (*a chunk* C_i) are being explored.

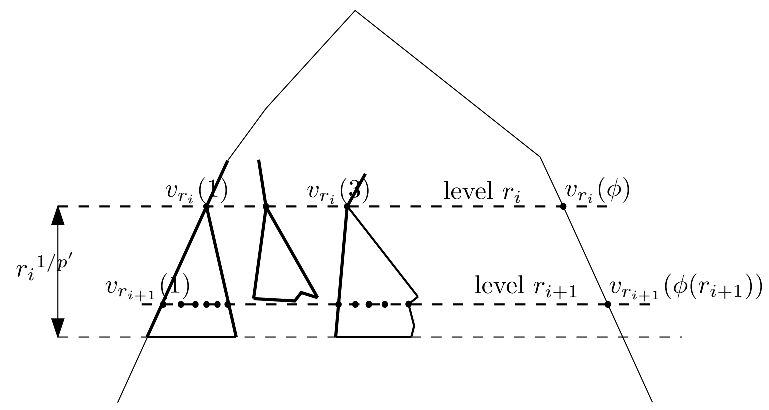

Fig. 2. One epoch of the *KE-2* algorithm

The following lemma demonstrates an idea of the algorithm and shows that the exploration does progress and additionally shows that the number of epochs is small.

Lemma 1. *The* KE-2 *terminates after executing $\mathcal{O}(D^{1-1/2p})$ epochs, where $p = p(T)$ is the density of T.*

Proof. We start by showing a property of some sequence a_i which suitable describes the behavior of our algorithm. Then using this property we show the upper bound on the number of epochs. We show that for $a_0 = 0, a_1 = 1$ and $a_{i+1} = a_i + \frac{1}{2} \cdot {a_i}^{1/2p}$ we have

$$a_i \geq \left(\frac{i}{6}\right)^{2p/(2p-1)} \tag{4}$$

Indeed, by induction we have the lower bound for $a_{i+1} = a_i + \frac{1}{2} \cdot a_i^{1/2p}$ of

$$(i/6)^{2p/(2p-1)} + \frac{1}{2} \cdot \frac{i^{1/(2p-1)}}{6^{1/(2p-1)}} \geq (i/6)^{2p/(2p-1)} + \frac{3 \cdot i^{1/(2p-1)}}{6^{2p/(2p-1)}} \geq \frac{(i+3)i^{1/(2p-1)}}{6^{2p/(2p-1)}} \geq$$

$$\geq \frac{(i+1)(i+1)^{1/(2p-1)}}{6^{2p/(2p-1)}} \geq \left(\frac{i+1}{6}\right)^{2p/(2p-1)} .$$

The main point of the above reasoning is that for $0 < \alpha \leq 1$, $i \geq 0$

$$\frac{i+3}{i+1} \geq \left(\frac{i+1}{i}\right)^{\alpha} .$$

Now let us observe that $r_i \geq a_i$. Therefore, by finding j s.t. $a_j \geq D$ we can argue that j epochs suffice to explore the whole tree. This is the case for $j = (D \cdot 6)^{1-1/2p}$ (we use (4) to show that) and thus at most $\mathcal{O}(D^{1-1/2p})$ epochs are executed. □

The value of the local variable p' at the beginning of the epoch E_i is the same for all robots and defines the value of p_i. By the sequence p_i our distributed algorithm estimates the upper bound for the density of the tree it explores. The lines 6–15 are *a turn* of the algorithm. Each epoch E_i consists of many turns during which subtrees in the chunk C_i are being explored.

All lines of the code (except one: the traversing according to *MyDFS*) are executed by all robots placed in the root of the tree. Then the local communication needed to compute e.g. ϕ and a new value of p' is granted. This emphasizes the locality and distributed property of our algorithm.

We now show that an execution of an epoch takes a small number of parallel steps for the *KE-2* algorithm.

Lemma 2. *The epoch E_i $(i \geq 1)$ of* KE-2 *terminates after*

$$\mathcal{O}\left(\frac{\phi(r_i)}{k} \cdot r_i \cdot [1 + \log(p_{i+1}/p_i)]\right)$$

time steps, after completely exploring the chunk C_i.

Proof. The epoch starts based on level r_i of the tree T, where exactly $\phi(r_i)$ subtrees $T_1(h), \ldots, T_{\phi(r_i)}(h)$ are rooted, where

$$T_j(h) = T_{v_{r_i}(j)}(h) \ .$$

Within one epoch, turn by turn, these subtrees are explored by our algorithm. During this execution the value of h changes. Initially $h = r_i^{1/p_i}$ and it decreases until it reaches $r_i^{1/p_{i+1}}$ at the end of the epoch.

The group of k robots starts the exploration from subtrees rooted at nodes with the smallest IDs. Since p_i is doubled during the exploration, the *repeat-until* loop, inside of the turn, terminates after $1 + \log(p_{i+1}/p_i)$ passes. Indeed, since $p_{i+1} \leq p$ the algorithm will reach some value, i.e. p_{i+1}, which is sufficient to explore all $T_v(r^{1/p_{i+1}})$ by *MyDFS* and then no *RISE* flag is set.
For each j and $p_i \leq p' \leq p_{i+1}$ the *MyDFS*$(v_{r_i}(j), r_i, p')$ algorithm takes at most $11r_i$ steps. This implies that one turn takes $\mathcal{O}(r_i)$ time steps. Moreover, there are exactly $\lceil \phi(r_i)/k \rceil$ turns during an epoch.

This proves that during the epoch E_i the algorithm explores the chunk C_i in time $\mathcal{O}\left(\frac{\phi(r_i)}{k} \cdot r_i \cdot [1 + \log(p_{i+1}/p_i)]\right)$ and eventually places all robots in the root. □

The following lemma describes the running time of the *KE-2* algorithm.

Lemma 3. *The* KE-2 *explores the tree T in time*

$$\mathcal{O}\left(\frac{\log p}{k} \sum_{i \geq 0} \phi(r_i) r_i\right) \ .$$

Proof. By Lemma 2, the epoch E_i needs at most $\phi(r_i)/k \cdot r_i \cdot [1 + \log(p_{i+1}/p_i)]$ time steps. Using $\frac{p_{i+1}}{p_i} \le 2p$ and summing up over all epochs we have

$$\sum_{i\ge 0} \frac{1}{k}\phi(r_i) \cdot r_i \cdot [1 + \log(p_{i+1}/p_i)] \le \frac{1}{k}\sum_{i\ge 0}\phi(r_i)\cdot r_i \cdot 2\log p$$

□

Now we compare the time of our online algorithm to the time the optimal offline algorithm would need for the same tree T. To show this we need to know the running time of the optimal algorithm.
In Lemma 4 we show a lower bound for running time of all algorithms exploring T. We describe this bound by using the sequence r_i and p_i, both defined by an execution of *KE-2*.

Lemma 4. *The optimal algorithm needs*

$$\Omega\left(\frac{1}{k}\sum_{i\ge 0}\phi(r_i){r_i}^{1/p_i}\right)$$

time steps.

Proof. Given the sequence r_i of levels of tree and the sequence p_i defined by the *KE-2*, define the set I_i of levels of T as follows

$$I_i = \left[r_i + \frac{1}{2}\cdot {r_i}^{1/p_{i+1}},\ r_i + {r_i}^{1/p_{i+1}}\right]$$

and let $\mathcal{I} = \bigcup_i I_i$ be the sum of these sets of levels. The levels I_i do not overlap since

$$r_{i+1} + \frac{1}{2}r_{i+1}^{1/p_{i+2}} > r_i + r_i^{1/p_{i+1}}$$

and thus the overall number of nodes at levels described by $\mathcal{I}$ is a lower bound on number of nodes in the tree T.

Let us count how many nodes n_i there are on levels contained in I_i. We know that $r_{i+1} \in I_i$ and that $\forall_{j\in I_i}(\phi(r_{i+1}) \le \phi(j))$. Then we have $n_i \ge 1/2 \cdot \phi(r_{i+1}) \cdot {r_i}^{1/p_{i+1}}$ and given $r_i \ge r_{i+1}/2$ we get

$$n_i = \Omega\left(\phi(r_{i+1})\cdot {r_{i+1}}^{1/p_{i+1}}\right)$$

so there are at least $\Omega\left(\sum_i \phi(r_i){r_i}^{1/p_i}\right)$ nodes in the tree.

A group of k robots needs at least s/k time steps to explore a tree with s nodes, which proves that even the optimal algorithm needs the time claimed in the lemma. □

We can prove now the competitive ratio of *KE-2* in the following lemma.

Lemma 5. *The* KE-2 *achieves competitive ratio of* $\mathcal{O}(\log p \cdot D^{1/2p} \cdot D^{1-1/p})$

Proof. First, we show that for any non-decreasing sequence $\{y_i\}$ and any sequence $\{x_i\}$ and for any $0 < \alpha < 1$ we have

$$\frac{\sum_{i=1}^{m} x_i y_i}{\sum_{j=1}^{m} x_i {y_i}^{\alpha}} \le {y_m}^{1-\alpha} \ . \tag{5}$$

To prove that, we notice

$$\frac{1}{y_m^{1-\alpha}} \cdot \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i \cdot y_i^{\alpha}} \le 1 \ .$$

Indeed, for all $0 \le i \le m$ we have $y_i^{1-\alpha} \le y_m^{1-\alpha}$ and we can upper-bound this term by

$$\frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i \cdot y_i^{\alpha} y_m^{1-\alpha}} \le \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i y_i^{\alpha} y_i^{1-\alpha}} \le 1 \ .$$

In the remaining part of the proof we lay $\alpha = 1/2p$, $y_i = r_i$ and $x_i = \phi(r_i)$ in (5).

We notice that ${r_i}^{1/p_i} \le D^{1/2p}$ and combine the results of Lemma 3 and 4 to provide the bound for competitive ratio σ.

$$\sigma \le \mathcal{O}\left(\log(4p) \cdot \frac{\sum_i \phi(r_i) r_i}{\sum_i \phi(r_i) {r_i}^{1/p_i}}\right) \le \mathcal{O}\left(\log p \cdot D^{1-1/2p}\right)$$

□

Now we present an algorithm which is a small modification of *KE-2*. Unlike the former, it will search for the value of density more accurately (not binary) and thus it will be a bit slower. The *KE-1* algorithm arises by replacing in Algorithm 1 the 9-th line of code $p' \leftarrow 2 \cdot p'$ by the new line $p' \leftarrow p' + 1$.

The proof of the performance of *KE-1* is quite the same as the proof of *KE-2* and thus we omit the detailed proof here. The *KE-1* is a distributed and online algorithm and achieves the following competitive ratio

Corollary 1. *The* KE-1 *is $\mathcal{O}(p \cdot D^{1-1/p})$-competitive.*

Proof. For the *KE-1* the local variable p' is increased only by one and thus epoch E_i takes

$$\mathcal{O}\left(\frac{\phi(r_i)}{k} \cdot r_i \cdot [1 + p_{i+1} - p_i]\right)$$

time steps. Summing up over all epochs we have $\mathcal{O}\left(p/k \cdot \sum_i \phi(r_i) r_i\right)$ time steps until tree is completely explored by *KE-1*. Since p_i approximates the value of $p(T)$ more accurately, we can use the lower bound for the time of optimal algorithm by Lemma 4, to obtain the competitive ratio of $\mathcal{O}\left(p \cdot D^{1-1/p}\right)$. □

We have now two algorithms *KE-1* and *KE-2* whose performance depends on the ratio between p and D. The first one is better for trees with a small density (comparing to D), and the second is better for a small – comparing to the density – height of the tree.

To take advantages of these algorithms and avoid drawbacks, we define the *KE-Clever* in the following way. The robot with the ID 1 (*a referee*) does not move but only waits in the root and serves as a relay station for the communication. It maintains the status of an exploration. Let now the first half of the remaining group of robots (the subgroup A) execute the *KE-1* and the other half (the subgroup B) the *KE-2* algorithm.

The *KE-Clever* terminates, after one of the group reports a completion of the exploration. The group A needs at most $\mathcal{O}\left(p \cdot D^{1-1/p}\right) \cdot C_{\text{OPT}}$ time and the group B needs $\mathcal{O}\left(\log p \cdot D^{1/2p} \cdot D^{1-1/p}\right) \cdot C_{\text{OPT}}$ time, which leads to the following theorem:

Theorem 1. *The* KE-Clever *achieves the competitive ratio of*

$$\mathcal{O}\left(D^{1-1/p} \cdot \min\left\{p, \log p \cdot D^{1/2p}\right\}\right)$$

for an arbitrary tree T, where D is the height of T and p the density.

2.3 The *KE-Oblivious* Algorithm

The algorithms *KE-1* and *KE-2* estimate the density value in a different way, but unfortunately, in the both cases the time needed for searching the proper value reflects unfavorably in their performance. Let us assume that the density $p(T)$ of the tree is known beforehand. Then, replacing in Algorithm 1 the 2-nd line of code $p' \leftarrow 1$ by the new line $p' \leftarrow p(T)$ we define a new algorithm called *KE-Oblivious*.

Theorem 2. *If the density parameter p for the tree is arbitrary but known before the exploration, then the* KE-Oblivious *has competitive ratio of*

$$\mathcal{O}\left(D^{1-1/p}\right)$$

for an arbitrary tree T, D in height.

Proof. Since the density is known in advance, there is no need for approximation. In fact, the local variable p' never changes during the execution of the algorithm. This results in only $\mathcal{O}\left(\frac{\phi(r_i)}{k} \cdot r_i\right)$ time steps needed for an epoch E_i. Following the same approach as in Lemma 3 and 4 we obtain an improved competitive ratio of $\mathcal{O}\left(D^{1-1/p}\right)$. □

3 Conclusions

We have presented two distributed online algorithms which explore the unknown tree starting from the root. Assuming global communication (or allowing landmarks in nodes) one can combine by a simple trick the *KE-Clever* or *KE-Oblivious* with the algorithm described in [15], obtaining the competitive ratio equal to the *minimum* of ratios of each combined algorithm. It is enough to execute them both in parallel – each algorithm executed by the half of the group.

In the face of the best known lower bound of 2-1/k, the problem concerning an optimal online competitive ratio for trees is still open. And finally, does the bounded communication have an impact on that ratio at all?

References

1. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, New York, NY, USA (1998)
2. Rao, N., Kareti, S., Shi, W., Iyenagar, S.: Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410 (1993)
3. Frederickson, G., Hecht, M., Kim, C.: Approximation algorithms for some routing problems. SIAM Journal on Computing **7** (1978) 178 – 193
4. Averbakh, I., Berman, O.: Minmax p-traveling salesmen location problems on a tree. Annals of Operations Research **110** (2002) 55 – 68
5. Dynia, M., Korzeniowski, M., Schindelhauer, C.: Power-aware collective tree exploration. In Grass, W., ed.: Proceedings of ARCS'06. LNCS 3894, Springer Verlag (2006) 341 – 351
6. Even, G., Garg, N., Könemann, J., Ravi, R., Sinha, A.: Min-max tree covers of graphs. Operations Research Letters **32** (2004) 309 – 315
7. Averbakh, I., Berman, O.: (p - 1)/(p + 1)-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective. Discrete Applied Mathematics **75** (1997) 201 – 216
8. Averbakh, I., Berman, O.: A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree. Discrete Applied Mathematics **68** (1996) 17 – 32
9. Bender, M., Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. In: Proc. FOCS 1994. (1994) 75 – 85
10. Bender, M., Fernández, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: exploring and mapping directed graphs. In: Proc. 30th Symp. Theory of Computing, ACM (1998) 269 – 278
11. Panaite, P., Pelc, A.: Exploring unknown undirected graphs. In: SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1998) 316 – 322
12. Awerbuch, B., Betke, M., Rivest, R., Singh, M.: Piecemeal graph exploration by a mobile robot. Information and Computation **152** (1999) 155 – 172
13. Dessmark, A., Pelc, A.: Optimal graph exploration without good maps. In: Algorithms - ESA 2002: 10th Annual European Symposium. Volume 2461., Springer (2002) 374 – 386
14. Fleischer, R., Trippen, G.: Exploring an unknown graph efficiently. In: 13th Annual European Symposium. Volume 3669., Springer (2005) 11 – 22
15. Fraigniaud, P., Gasieniec, L., Kowalski, D., Pelc, A.: Collective tree exploration. In: Proc. LATIN 2004. Volume 2976. (2004) 141–151
16. Fraigniaud, P., Ilcinkas, D., Rajsbaum, S., Tixeuil, S.: Space lower bounds for graph exploration via reduced automata. In: Structural Information and Communication Complexity: 12 International Colloquium. Volume 3499. (2005) 140 – 154