Why Robots Need Maps*

Miroslaw Dynia[1], Jakub Łopuszański[2], and Christian Schindelhauer[3]

[1] DFG Graduate College "Automatic Configuration in Open Systems", University of Paderborn, Germany,
mdynia@uni-paderborn.de
[2] Institute of Computer Science, University of Wroclaw, Poland,
jakub.lopuszanski@ii.uni.wroc.pl
[3] Computer Networks and Telematics, University of Freiburg, Germany,
schindel@informatik.uni-freiburg.de

Abstract. A large group of autonomous, mobile entities e.g. robots initially placed at some arbitrary node of the graph has to jointly visit all nodes (not necessarily all edges) and finally return to the initial position. The graph is not known in advance (an online setting) and robots have to traverse an edge in order to discover new parts (edges) of the graph. The team can locally exchange information, using wireless communication devices.
We compare cost of the online and optimal offline algorithm which knows the graph beforehand (competitive ratio). If the cost is the total time of exploraiton, we prove the lower bound of $\Omega(\log k/\log\log k)$ for competitive ratio of any deterministic algorithm (using global communication). This significantly improves the best known constant lower bound. For the cost being the maximal number of edges traversed by a robot (the energy) we present an improved $(4-2/k)$-competitive online algorithm for trees.

1 Introduction and Our Results

We are interested in the issue of coordination of a team of k autonomous robots. We would like the team to be driven by a distributed algorithm stored in the local memory and executed using the local computational power of a robot. The team's goal is to jointly visit all nodes of an unknown, but labeled graph G. To let the team cooperate we must allow it to exchange information about new findings and agree upon the strategy of the exploration. We can allow full communication scheme (global communication), yet it is more realistic to allow only local communication. Robots are equipped with wireless radio devices with a bounded communication radius which allows only the neighboring robots to communicate.

* 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 MNiSW grant number N206 001 31/0436, 2006-2008

The team is initially placed in a node of the unknown graph G modeling the network or e.g. an unknown terrain, where nodes correspond to the interesting locations, and edges model the accessibility between locations. Additionally, we assume that all edges and nodes of G are labeled and thus can be locally distinguished by a robot. A goal for the robots is to jointly visit all nodes of the graph and finally return to the initial node. We consider two cost models. In the first model we assume the cost measure to be the total time of the exploration where in the second model it is the maximal energy (number of edges) used by a robot.

It is clear that knowing the graph beforehand (*offline setting*), the team would agree on the best strategy before the exploration and then fully explore the graph without using communication at all. Intuitively, the cost of such an exploration should be smaller than the cost needed in the *online setting* where the graph is not known in advance.

The *competitive ratio* is the ratio between the cost of the online and the optimal offline algorithm. Competitive ratio of 1 would mean that robots can efficiently explore even though the "map of the graph" is not known. In fact, for the time model we show that this ratio is significantly larger even assuming the global communication (see Sect. 3). We show the lower bound of $\Omega(\log k / \log \log k)$ for the competitive ratio of any deterministic graph exploration algorithm using k robots. This is a significant improvement comparing to the constant factor bounds known so far. For the energy model (Sect. 4) we show the $(4 - 2/k)$-competitive algorithm which explores trees and uses strictly local communication (in fact robots communicate only in the root of tree).

2 Prior and Related work

There are many results (e.g. [1–6]) concerning exploration of a graph using small number of robots. Authors of [5] present strategies for a robot which has to traverse all edges minimizing the number of edge traversals. They bound competitive ratio (an overhead related to the lack of topology's knowledge) of their algorithms for several classes of graphs. Authors of [2] show a strategy for two robots to explore (in polynomial time) all nodes of an unlabeled, strongly connected, directed graph.

The real impact of robot's cooperation can be observed by studying the algorithms which use larger number of robots ($k > 2$). Robots can collectively perform many tasks (e.g. black-hole search [7, 8] or rendezvous [9]) but here we focus on the problem of an exploration. Dealing with a group of robots, there are many coordination problems e.g. gathering or pattern formation [10] which a team might exercise during the exploration of an unknown terrain (or unlabeled graph like in [2, 11]). Those problems might arise from the sensor inaccuracy, odometry error related to the movement or from some computational problems. However, in many publications this is overcome assuming that either the exploration concerns the network or the terrain is represented by the labeled graph in which these problems do not occur.

The problem of collective tree exploration is addressed in [12]. Authors present the lower bound of $2 - 1/k$ for competitive ratio of an arbitrary exploration algorithm using k robots. Additionally, they prove that if no communication is allowed, it is not possible to explore efficiently (competitive ratio $\Omega(k)$). Their online algorithm for tree exploration uses global communication and achieves competitiveness of $O(k/\log k)$. Additionally, in [13] we present online algorithms for exploration of so called sparse trees and e.g. for trees D in height, which can be embedded in 2-dimensional grids, the algorithm achieves competitive ratio of $O(\sqrt{D})$. The problem with the cost model related to the maximal energy used by a robot is addressed in [14]. Authors present the lower bound of 3/2 and distributed 8-competitive algorithm for trees which uses only local communication.

In [15] the group of simple robots (could be represented by the primitive final automata) fills the integer grid subject to minimize the makespan. Initially, the robots are standing outside the grid (by so called doors) and can consecutively enter the grid, with the additional assumption that only one robot can occupy one node. They develop optimal solution for the single-door case and $O(\log(k+1))$-competitive algorithm for multi-door case.

3 The Time Model

Consider an arbitrary graph $G = (E, V)$ with a distinguished node $s \in V$. The team of k autonomous mobile robots starts in s, visits all nodes of G and finally returns to s. It takes one time step for the robot to traverse an edge and since there are many synchronized robots, there might be many edges which are traversed in parallel by the team during one time step.

In this chapter we focus on the total time of such an exploration and furthermore we compare it to the time needed by the optimal offline algorithm. We show that for each deterministic algorithm there exists a tree-like graph which cannot be efficiently explored. Although there is a local communication granted for robots we show that having even global communication does not help if we do not know the map in advance. This means that in the worst case no online algorithm can explore efficiently, if compared to the time needed by the optimal offline algorithm.

First in the Sect. 3.1 we introduce the Jellyfish tree which is used in Sect. 3.2 to prove the lower bound of $\Omega(\log k/\log\log k)$ for competitive ratio of an arbitrary deterministic exploration algorithm $\mathcal{A}$.

3.1 The Jellyfish tree

We define a class of trees which contains the lower bound tree for any arbitrary deterministic algorithm. Assume $t > k$ and take some permutation σ of set 1 through k. *Jellyfish tree* $J(k, t, \sigma)$ consists of k subtrees (*tentacles*) numbered from 1 trough k, connected by the root (see Fig. 1).

The tentacle consists of a *poison* of a certain size which is attached to the single path of the length t. Each level of poison (but the first one) consists of t nodes, all connected to the *main node* of the previous level. The main node is the one which was visited by any robot as the last one on this level. Therefore, all nodes on the level l have to be visited before any other node on the level $l+1$ is visited. This also means that any algorithm using $k' \leq k$ robots traversing one tentacle of size s_i needs at least $t + 2\frac{t \cdot l}{k'}$ time steps to reach the main node on level l. However, if the configuration of the main nodes is known, it takes only $t + 2l$ to do the same. The complete exploration of this tentacle needs at least $2t + 2\frac{t \cdot s_i}{k'}$ if main nodes are not known.

The size of a poison contained in the i-th tentacle is defined by the function

$$s_{\sigma(i)} := \left\lceil \frac{k}{\log k} \cdot \frac{1}{i} \right\rceil ,$$

where the permutation σ allows to rearrange the order of poisons. Later, in Sect. 3.2 we define the permutation $\sigma_{\mathcal{A}}$ for the algorithm $\mathcal{A}$ in such a way, that the sizes of poisons are in the inconvenient configuration for the algorithm.

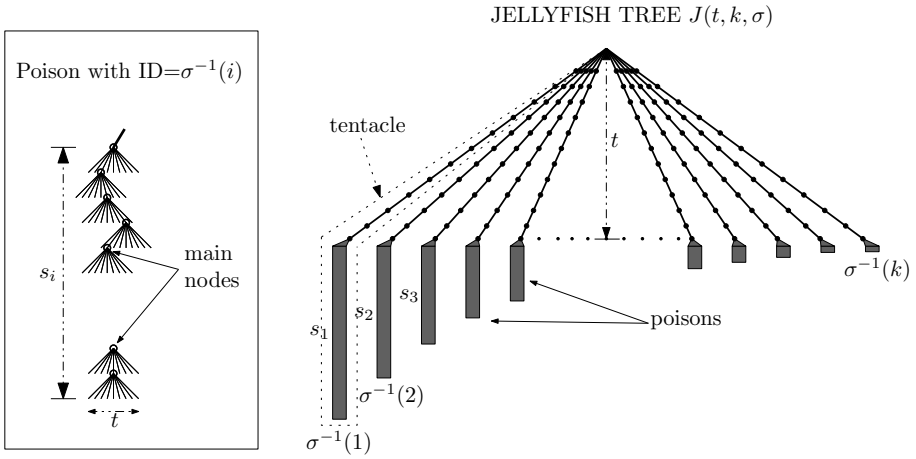

Fig. 1. Definition of a poison and the Jellyfish tree $J(t, k, \sigma)$.

3.2 Online Analysis

For an arbitrary (yet fixed) algorithm $\mathcal{A}$ we define a particular lower bound tree $J(t, k, \sigma_{\mathcal{A}})$ by defining a permutation $\sigma_{\mathcal{A}}$ and taking an arbitrarily large t (it must be at least $t > k$). The configuration of the main nodes within $J(t, k, \sigma_{\mathcal{A}})$ is also defined by $\mathcal{A}$. The permutation $\sigma_{\mathcal{A}}$ is constructed in such a way, that it

maximizes the ratio between the exploration time of $\mathcal{A}$ and an optimal offline algorithm on the same tree.

We start with the observation that being independent on the permutation and other parameters of the Jellyfish tree, the offline exploration is easy.

Lemma 1. *Assuming that the tree is known beforehand (i.e. $t > k$ and σ are known), there exists the algorithm with k robots which explores $J(t, k, \sigma)$ in $O(t)$ steps.*

Proof. The graph $J(t, k, \sigma_A)$ is $h := t + k/\log k$ in height and has at most

$$2tk + (tk/\log k) \cdot \sum_{i=1}^{k} 1/i = O(tk)$$

nodes. Therefore we have $h = O(t)$ and $n/k = O(t)$ and using e.g. the approximation algorithm from [14] we can get the algorithm exploring in time $O(t)$. □

To prove a bad performance of the algorithm $\mathcal{A}$ we study the performance of the similar algorithm which works in a synchronized rounds. We define a class of synchronized algorithms (see Alg. 1) which explores the given Jellyfish tree. The particular algorithm within this class is fully defined by the sequence $f_i^{(r)}$ $(1 \le i \le k,\ r \ge 1)$ which dictates the distribution of robots among the tentacles within the round. Therefore, for each round r we require $f_i^{(r)} \ge 0$ and additionally

$$\sum_{i=1}^{k} f_i^{(r)} = k \ .$$

During a round, the value $f_i^{(r)}$ defines the size of the subgroup assigned to explore the i-th tentacle. The exploration within the assigned tentacles is carried out in parallel, and at the end of the round, all robots meet in the root to make a new assignment for the next round.

Define $a_j^{(r)} := 1$ for $j \le f_1^{(r)}$ and otherwise $a_j^{(r)} := q$, where q fulfills

$$\sum_{i=1}^{q-1} f_i^{(r)} < j \le \sum_{i=1}^{q} f_i^{(r)} \ .$$

If the j-th robot goes to the tentacle $a_j^{(r)}$ it results in the distribution described by $f_i^{(r)}$. The sequence $f_i^{(r)}$ is known to each robot and thus the team can easily agree upon the assignment.

Algorithm 1 Synchronized Algorithm

Require: $id \leftarrow$ robot's ID
  $r \leftarrow 1$ *// round counter*
  while (J is not yet explored) **do**
    $j \leftarrow a_{id}^{(r)}$ *// the assigned tentacle*
    $v \leftarrow$ the furthest explored main node in the j-th tentacle
    move to v (using a simple path in tree)
    $l \leftarrow 0$
    while ($l < f_j^{(r)}$ and j-th tentacle is not yet explored) **do**
      collectively visit all t nodes connected by an edge to v
      $v \leftarrow$ the next main node connected to v
      $l \leftarrow l + 1$
    end while
    return to the root (using a simple path in tree)
    $r \leftarrow r + 1$
  end while

Collective exploration carried out in the second 'while' loop is executed by the team of $k' = f_j^{(r)}$ robots initially located in the main node v. There are t edges outgoing from v (one of them is connected to the main node of the further level). The team spreads itself uniformly among those edges, end explores it in parallel. Since there are more edges than robots, this must be repeated $\lceil t/k' \rceil$ times until all edges on this level are explored. The main node on the further level is known and the team moves there to explore further levels. The loop breaks when there are $f_j^{(r)}$ additional levels explored, which takes

$$k' \cdot (2 \cdot \lceil t/k' \rceil + 1) - 1 \leq 5t$$

time steps.

Traveling to the first main node takes t steps, traveling along the already explored main nodes inside of a poison takes also at most t steps (as we have $s_i < k < t$). As we have seen, the further exploration takes at most $5t$ steps and return to the base takes at most $2t$ steps. This can be summarized in the following lemma.

Lemma 2. *The r-th round of the $\mathcal{A}_{\text{sync}}$ algorithm takes at most $9t$ time steps and there are $f_j^{(r)}$ new levels explored in the j-th tentacle (if the exploration is not finished earlier). The configuration of the main nodes does not influence this performance.*

Now we show how to "synchronize" an arbitrary online algorithm $\mathcal{A}$ exploring the $J(k, t, \sigma)$. We find the matching algorithm within the class of synchronized algorithms by splitting the execution of $\mathcal{A}$ in the time intervals $T_1, T_2, \ldots, T_R$ each t in length (the last interval might be shorter). For $1 \leq r \leq R$ define $f_i^{(r)}$ as the number of robots which visit the poison contained in the i-th tentacle during time interval T_r. If $k_{\text{lazy}} \leq k$ robots visit no poison during this interval, increase

$f_i^{(r)}$ for an arbitrary i by k_{lazy}. Observe that no robot can visit two poisons during one time interval, and thus for each r we have $\sum_i f_i^{(r)} = k$. We say that the deterministic algorithm $\mathcal{A}$ defines the sequence $f_i^{(r)}$. If the synchronized algorithm uses the sequence $f_i^{(r)}$ defined by the $\mathcal{A}$ algorithm, then it is called a *synchronized algorithm based on* $\mathcal{A}$ and we denote it by $\mathcal{A}_{\text{sync}}$.

Lemma 3. *If $\mathcal{A}$ explores the jellyfish tree J in time T then $\mathcal{A}_{\text{sync}}$ explores the same tree in at most $\lceil T/t \rceil$ rounds.*

Proof. The $\mathcal{A}$ algorithm consecutively uses at most $f_j^{(1)}, f_j^{(2)}, \ldots f_j^{(R)}$ robots to explore the j-th tentacle in the time interval $T_1, \ldots T_R$ (where $R = \lceil T/t \rceil$). Let s be the size of the poison in this tentacle. The poison contains st edges and each has to be traversed by a robot thus

$$\sum_{r=1}^{R} (t \cdot f_j^{(r)}) \geq t \cdot s \ ,$$

where $t \cdot f_j^{(r)}$ is a maximal number of edges which can be traversed in t steps by $f_j^{(r)}$ robots.

In $\mathcal{A}_{\text{sync}}$ algorithm there are exactly $f_j^{(r)}$ robots exploring the j-th tentacle in the r-th round, and they make progress of $f_j^{(r)}$ additional levels. Summing up all rounds we have $\sum_{r=1}^{R} f_j^{(r)}$ levels explored. We know that this sum is at least s, and thus $\mathcal{A}_{\text{sync}}$ completely explores this tentacle in R rounds.

Since j was arbitrarily chosen this holds for an arbitrary tentacle, which means that the whole jellyfish tree will be also explored by $\mathcal{A}_{\text{sync}}$ in $R = \lceil T/t \rceil$ rounds. □

Let $\mathcal{C}_{\mathcal{A}}[J]$ and $\mathcal{C}_{\mathcal{A}_{\text{sync}}}[J]$ be the time the algorithms $\mathcal{A}$ and $\mathcal{A}_{\text{sync}}$ need for the exploration of J. By Lemma 3 the $\mathcal{A}_{\text{sync}}$ algorithm explores J in $\lceil \mathcal{C}_{\mathcal{A}}[J]/t \rceil$ rounds, and each round takes at most $9t$ time steps (Lemma 2). We have $\mathcal{C}_{\mathcal{A}_{\text{sync}}}[J] \leq 9t \cdot \lceil \mathcal{C}_{\mathcal{A}}[J]/t \rceil \leq 18 \cdot \mathcal{C}_{\mathcal{A}}[J]$. The synchronization results in only a constant factor increase in the time of an algorithm exploring an arbitrary Jellyfish tree.

Lemma 4. *Let $\mathcal{A}$ be the algorithm using k robots to explore $J := J(k, t, \sigma)$ Jellyfish tree (with arbitrarily σ and t). There holds*

$$\mathcal{C}_{\mathcal{A}_{\text{sync}}}[J] \leq 18 \cdot \mathcal{C}_{\mathcal{A}}[J]$$

where $\mathcal{A}_{\text{sync}}$ is a synchronized algorithm based on $\mathcal{A}$.

Now, we show how the adversary reorders the sizes of poisons for $\mathcal{A}_{\text{sync}}$ to make the exploration harder. If the online exploration is hard for $\mathcal{A}_{\text{sync}}$, it is also hard for $\mathcal{A}$ (by Lemma 4). Consider one round of a synchronized algorithm $\mathcal{A}_{\text{sync}}$ and assume that the decision on the size of a group assigned to a particular

poison for this round is already made. The adversary orders the poison's size in such a way that large groups of robots get small-sized poisons and small groups of robots get large-size poisons. More precisely, if we sort groups of robots in an increasing order of their sizes, the (still unexplored) assigned poisons are sorted in a decreasing order of their sizes.

If after this round some poison with id i gets fully explored and then $\sigma(i)$ gets fixed. All remaining (not fully explored) poisons are subject to reordering done at the beginning of the next round. As the adversary reorders sizes of (not fully explored) poisons, it partially defines permutation $\sigma_{\mathcal{A}}$, which gets fully defined at the end of the exploration.

In Lemma 5, we show that the adversarial order of poison's size increases the exploration time for the algorithm $\mathcal{A}_{\text{sync}}$ based on algorithm $\mathcal{A}$. We prove that the progress of an exploration done by $\mathcal{A}_{\text{sync}}$ (measured by the deepest visited node) is small, assuming that the Jellyfish tree is constructed on the permutation $\sigma_{\mathcal{A}}$ defined above.

Lemma 5. *At the end of the r-th round of algorithm $\mathcal{A}_{\text{sync}}$ no node at distance greater than $y^{(r)} := t + \lceil (2\log k)^r \rceil$ from the root is visited (where k is large, at least $\log k \geq 5$).*

Proof. Assume that v is visited in the first round and it is at distance $t + \lceil 2\log k \rceil + 1 \geq t + 11$ from the root. The node v lies within the poison of the j-th tentacle and there are $f_j^{(1)} \geq 11$ robots exploring this poison of size $s \geq 11$ in this round. There are at least $k/2$ tentacles which contain a poison smaller than 11 and each of them is also explored by at least 11 robots (adversarial order of poison's size). This gives $11 \cdot k/2 > k$ robots exploring the smallest tentacles in this round.

Assume that at the beginning of the round $r > 1$ (at the end of the round $r-1$) there is no visited node which lies deeper than at distance $y^{(r-1)}$. Furthermore, assume that all nodes up to the distance $y^{(r-1)}$ are already visited. Now the algorithm declares the distribution $f_i^{(r)}$ for this round and the adversary sorts tentacles in a reversed order of their sizes to an order of $f_i^{(r)}$.

Assume that there is a node v which is visited before the end of this round and lies at the distance $y^{(r)}+1$ from the root. The node v is contained in the j-th tentacle for which $\mathcal{A}_{\text{sync}}$ has had many robots exploring it during the round r. Certainly, $\mathcal{A}_{\text{sync}}$ needs much "exploration power" to rapidly explore many levels during one round. To explore additional $y^{(r)} - y^{(r-1)}$ levels of the j-th tentacle, algorithm $\mathcal{A}_{\text{sync}}$ needs $f_j^{(r-1)} \geq y^{(r)} - y^{(r-1)}$ robots, and therefore we have

$$f_j^{(r-1)} \geq (\log k)^{r-1} \cdot (2\log k - 2) \ .$$

For $i = \lceil k/\log k \cdot 1/(2\log k)^{r+1} \rceil$ we have $s_i \leq y^{(r)}$ and for $i' = \lfloor k/\log k \cdot 1/(2\log k)^r \rfloor$ we have $s_{i'} \geq y^{(r-1)}$ and so there are at least

$$i' - i \geq \frac{k}{\log k \cdot (2\log k)^{r-1}} \cdot (1 - 1/\log k)$$

tentacles which end between levels $y^{(r-1)}$ and $y^{(r)}$. All those tentacles have at least $f_j^{(r-1)}$ robots assigned to them in this round (because the adversary has sorted the tentacles in a reversed order). This means that there were at least

$$f_j^{(r)} \cdot (i' - i) \geq k \cdot (2 - 4/5) > k$$

robots exploring tree in the r-th round. This contradicts the fact that $\mathcal{A}_{\text{sync}}$ uses only k robots to explore the Jellyfish tree. □

Using Lemma 5 we know that in the r-th round the algorithm $\mathcal{A}_{\text{sync}}$ makes only a small progress. If the largest poison (containing a poison of size $k/\log k$) is explored in the R-th round then $y^{(R)} \geq t + k/\log k$. It means that $(\log k)^R \geq k/\log k$ and so

$$R = \Omega\left(\frac{\log k}{\log\log k}\right) ,$$

where each round takes $\Theta(t)$ time steps.

The algorithm $\mathcal{A}$ is at most 18 times faster than $\mathcal{A}_{\text{sync}}$ by the exploration of $J(t, k, \sigma_{\mathcal{A}})$ (see Lemma 4) and thus $\mathcal{A}$ needs also $\Omega\left(t \cdot \frac{\log k}{\log\log k}\right)$ time steps. On the other hand, from Lemma 1 we know that $O(t)$ steps suffice to explore this tree. Since we have chosen $\mathcal{A}$ to be the arbitrary algorithm using k robots for the collective graph exploration we have proved the following theorem:

Theorem 1. *For every online exploration algorithm $\mathcal{A}$ using k robots there exists a graph for which the total time of the exploration is at least*

$$\Omega\left(\frac{\log k}{\log\log k}\right)$$

longer than the optimal time needed to explore this graph offline by k robots.

4 Algorithm for the Energy Model

In this section we consider a tree $T = (E, V)$ rooted at $v_0 \in V$ consisting of n uniform labeled edges and D in height measured by the number of edges on the longest path from v_0 to a leaf. All k robots with an unique ID drawn from the set $1, 2, \ldots, k$ are initially placed in v_0. Robots can communicate when they are in the same node and a goal of such a team is to jointly explore the unknown tree.

Assume that whenever a robot traverses an edge, it incurs a cost of one energy unit. We are interested in costs of the exploration defined as the maximal energy used by a robot. Once again we compare the cost of the online algorithm to the cost of the optimal offline algorithm to obtain competitive ratio. In [14] the lower bound of $3/2$ as well as 8-competitive online algorithm exploring tree using a team of k robots were shown. Here we improve this result by introducing $4-2/k$-competitive algorithm. This confirms that the energy model is strictly weaker

than the time model for which the first non-constant lower bound is presented in the previous section.

In the energy model robots do not care about an overall time of the exploration. In some situations halting and waiting for a new information may be more desirable for a robot than further exploration. This is exactly what happens in our algorithm. We have a group of k robots $(R_1, R_2, \ldots R_k)$ and during a round, there is only one robot which is active. All other robots are waiting in the root v_0 of the tree. The active robot first goes to the node where the robot active in the previous round has given up its exploration. Then it continues this exploration for a certain number of steps, and finally returns to the root v_0. The algorithm is described in details on the Fig. 2.

Algorithm 2 `PushDfs`

$h \leftarrow 1$
$r \leftarrow 1$
$e_i \leftarrow 0$ for all $1 \leq i \leq k$

while (T is not yet explored) **do**
  $id \leftarrow argmin\{e_i : 1 \leq i \leq k\}$
  R_{id} travels to v_{r-1}
  $e \leftarrow 0$
  repeat
    R_{id} follows a DFS step
    $e \leftarrow e + 1$
    $h \leftarrow$ height of the visited subtree
  until $(e \geq 2h)$
  $h_r \leftarrow h$
  $v_r \leftarrow$ actual position of R_{id}
  R_{id} returns to v_0 *//this takes at most h_r edges*
  $e_{id} \leftarrow e_{id} + |path(v_0, v_{r-1})| + 2h_r + |path(v_r, v_0)|$
  $r \leftarrow r + 1$
end while

The "while" loop corresponds to one round r, and there is only one robot R_{id} which moves during this round. Variable h_r denotes the height of the subtree visited by a robot in all rounds from 1 through r and the variable e_i holds the energy used so far by the i-th robot.

In the r-th round, the active robot first travels to the node v_{r-1} at which the previous robot stopped exploring (it takes at most h_{r-1} energy units). Then it continues to traverse consecutive DFS edges ("repeat" loop) until it collects $2h$ of it, where h is the height of the subtree visited by any robot (also the actual active one) so far. During this loop the robot traverses $2h_r$ edges and then finally returns to v_0 (which certainly takes at most h_r energy units). This gives us an upper bound of $h_{r-1} + 3h_r$ on the energy used by a robot during the round r. In the first round ($r = 1$) active robot is the first working robot ever, so it uses only

$0 + 3h_1$ energy units (we lay $h_0 = 0$). The last round q is perhaps shorter and it takes $h_{q-1} + w$ because a robot gets to the root already during the "repeat" loop and thus does not have to pay any extra costs for the return.

Let e_i be the energy of the robot R_i after completely exploring the tree and $E = \sum_{i=1}^{k} e_i$ be the total energy used by all robots. We have

$$E \leq 3h_1 + \sum_{r=2}^{q-1}(h_{r-1} + 3h_r) + h_{q-1} + w = 4 \cdot \sum_{r=1}^{q-1} h_r + w$$

as the upper bound for this energy. On the other hand we know that the robot active in the round $r \leq q-1$ has done exactly $2h_r$ steps of the DFS tour (the last one exactly w steps), which for the whole tree takes exactly $2n$ energy units. It must be that $(\sum_{i=r}^{q-1} 2h_r) + w = 2n$ and thus we have

$$E \leq 4n \leq 2k \cdot OPT$$

where OPT is the energy cost of the optimal offline exploration. Indeed, $OPT \geq 2n/k$ since there are n edges and each has to be traversed twice by at least one robot.

Let $e_{\min} = min\{e_i : 1 \leq i \leq k\}$ and $e_{\max} = max\{e_i : 1 \leq i \leq k\}$ be respectively the minimal and the maximal energy used by robots $R_{\min}$ and $R_{\max}$. One tour during a round takes at most $h_{q-1} + 3h_q \leq D + 3D = 4D$ energy units of an active robot. The active robot is chosen to be the one with the smallest energy used so far and therefore we have $e_{\max} - e_{\min} \leq 4D$. Certainly $OPT \geq 2D$, because there is at least one robot which has to reach the furtherest leaf at distance D and return to v_0. This results in the upper bound

$$e_{\max} - e_{\min} \leq 2OPT \ .$$

Knowing the span of values of the elements of the sequence e_i we can use the following upper bound on the maximal value

$$e_{\max} \leq \frac{\sum_{i=1}^{k} e_i - (e_{\max} - e_{\min})}{k} + (e_{\max} - e_{\min})$$

and therefore we obtain

$$e_{\max} \leq \frac{2k \cdot OPT}{k} + (1 - 1/k) \cdot 2OPT \leq (4 - 2/k) \cdot OPT \ .$$

This analysis can be slightly improved (at the cost of readability) but it also can be proved that the competitive ratio of this algorithm asymptotically converges to 4.

Theorem 2. *The* `PushDfs` *algorithm explores an arbitrary tree and obtains the competitive ratio of at most* $4 - 2/k$ *for the online energy model.*

5 Conclusions

We have presented the lower bound of $\Omega(\log k / \log\log k)$ for a competitive ratio in the time model of the exploration. This is a significant improvement over the recent $2 - 1/k$ lower bound presented in [12]. The best algorithms for trees achieve a competitiveness of $O(k/\log k)$ and $O(\sqrt{D})$ (for sparse trees) which leaves a wide area for further research. Moreover, our result proofs that the lack of a map is essentially harmful in the time related online graph exploration problem (and this remains even when we restrict ourself only to trees).

For the energy cost model there is an online algorithm with constant competitive ratio for trees. Using a simple algorithm the team of k robots can explore a tree using only $4 - 2/k$ times the energy of the offline solution. This does not match yet the lower bound of $3/2$ presented in [14].

References

1. Betke, M., Rivest, R.L., Singh, M.: Piecemeal learning of an unknown environment. In: Proc. of the 6th Annual ACM Conference on Computational Learning Theory (COLT 1993), Association for Computing Machinery (1993) 277–286
2. Bender, M., Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. In: Proc. of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994), IEEE Computer Society (1994) 75–85
3. Bender, M.A., Fernández, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: Exploring and mapping directed graphs. Information and Computation **176** (2005) 1–21
4. Fleischer, R., Trippen, G.: Exploring an unknown graph efficiently. In: Proc. of the 13th Annual European Symposium on Algorithms (ESA 2005), Springer Verlag (2005) 11–22
5. Dessmark, A., Pelc, A.: Optimal graph exploration without good maps. Theoretical Computer Science **326** (2004) 343–362
6. Gasieniec, L., Pelc, A., Radzik, T., Zhang, X.: Tree exploration with logarithmic memory. In: Proc. of ACM-SIAM Symp. on Discrete Algorithms (SODA 2007). (2007)
7. Dobrev, S., Flocchini, P., Kralovic, R., Ruzicka, P., Prencipe, G., Santoro, N.: Black hole search in common interconnection networks. Networks **47** (2006) 61–71
8. Dobrev, S., Flocchini, P., Prencipe, G., Santoro, N.: Searching for a black hole in arbitrary networks: optimal mobile agent protocols. In: Proc. of the 21st Annual Symposium on Principles of Distributed Computing (PODC '02). (2002) 153–162
9. Dessmark, A., Fraigniaud, P., Kowalski, D., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica **46** (2006) 69– 96
10. Sugihara, K., Suzuki, I.: Distributed algorithms for formation of geometric patterns with many mobile robots. Journal of Robotic Systems **13** (1996) 127–139
11. Das, S., Flocchini, P., Nayak, A., Santoro, N.: Distributed exploration of an unknown graph. In: Proc. of the Structural Information and Communication Complexity (SIROCCO 2005), Springer Verlag (2005) 99–114
12. Fraigniaud, P., Gasieniec, L., Kowalski, D., Pelc, A.: Collective tree exploration. Networks **48** (2006) 166–177

13. Dynia, M., Kutyłowski, J., Meyer auf der Heide, F., Schindelhauer, C.: Smart robot teams exploring sparse trees. In: Proc. of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS 2006), Springer Verlag (2006) 327–338
14. Dynia, M., Korzeniowski, M., Schindelhauer, C.: Power-aware collective tree exploration. In: Proc. of the 19th International Conference on Architecture of Computing Systems (ARCS 2006), Springer Verlag (2006) 341–351
15. Hsiang, T., Arkin, E., Bender, M., Fekete, S., Mitchell, J.: Algorithms for rapidly dispersing robot swarms in unknown environments. In: Proc. of the 5th International Workshop on Algorithmic Foundations of Robotics, Springer Berlin / Heidelberg (2002) 77– 94