Collective tree exploration

Pierre Fraigniaud[1⋆], Leszek Gąsieniec[2⋆⋆], Dariusz R. Kowalski[3,4⋆⋆⋆], and Andrzej Pelc[5†]

[1] CNRS-LRI, Université Paris-Sud, 91405 Orsay, France, *http://www.lri.fr/~pierre*.
[2] Department of Computer Science, The University of Liverpool, Liverpool L69 7ZF, UK, *E-mail: leszek@csc.liv.ac.uk.*
[3] Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, Saarbrücken, 66123 Germany.
[4] Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02-097 Warszawa, Poland, *E-mail: darek@mimuw.edu.pl.*
[5] Département d'informatique, Université du Québec en Outaouais, Hull, J8X 3X7, Canada, *E-mail: pelc@uqo.ca.*

Abstract. An n-node tree has to be explored by k mobile agents (robots), starting in its root. Every edge of the tree must be traversed by at least one robot, and exploration must be completed as fast as possible. Even when the tree is known in advance, scheduling optimal collective exploration turns out to be NP-hard. We investigate the problem of distributed collective exploration of unknown trees. Not surprisingly, communication between robots influences the time of exploration. Our main communication scenario is the following: robots can communicate by writing at the currently visited node previously acquired information, and reading information available at this node. We construct an exploration algorithm whose running time for any tree is only $O(k/\log k)$ larger than optimal exploration time with full knowledge of the tree. (We say that the algorithm has *overhead* $O(k/\log k)$). On the other hand we show that, in order to get overhead sublinear in the number of robots, some communication is necessary. Indeed, we prove that if robots cannot communicate at all, then every distributed exploration algorithm works in time $\Omega(k)$ larger than optimal exploration time with full knowledge, for some trees.

1 Introduction

A collection of robots (mobile agents), initially located at one node of an undirected connected graph, have to explore this graph and return to the starting point. The graph is explored if every edge is traversed by at least one robot. Every robot traverses any edge in unit time, and the time of collective exploration is the maximum time used by any robot from the group. It turns out that scheduling optimal collective exploration is NP-hard, even in the simplest case, when the explored graph is a tree and when it is known in advance. However, most often, exploration problems are studied in the case of unknown graphs (cf. [1, 6, 12, 14–17, 21]). This is also the approach adopted in the

⋆ Research supported by the Actions Spécifiques CNRS *Dynamo* and *Algorithmique des grands graphes*, and by the project *PairAPair* of the ACI *Masse de données*.
⋆⋆ Research partially supported by the EPSRC grant GR/R84917/01.
⋆⋆⋆ This work was done in part during this author's stay at the Research Chair in Distributed Computing of the Université du Québec en Outaouais, as a postdoctoral fellow. Research supported in part by KBN grant 4T11C04425.
† Research supported in part by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

present paper. We restrict attention to *trees* and, unlike in the above quoted papers, we consider exploration by *many* robots. The goal is to collectively explore the tree in the shortest possible time. Since the explored tree is not known in advance, a collective exploration algorithm can have different performance in different trees. In order to measure the quality of such an algorithm, we compare its performance to the performance of the optimal exploration algorithm which knows the tree in advance (recall that designing such an optimal exploration is NP-hard). A collective exploration algorithm $\mathcal{A}$ for k robots (working in unknown trees) is said to have *overhead* Q, if Q is the supremum of ratios $\mathcal{A}(k,T,r)/opt(k,T,r)$, where $\mathcal{A}(k,T,r)$ is the exploration time of tree T by algorithm $\mathcal{A}$, when robots start at node r, and $opt(k,T,r)$ is the optimal exploration time of T by k robots starting at r, assuming that T and r are known. The supremum is taken over all trees T and starting nodes r. Hence overhead is a measure of performance similar to competitive ratio for on-line algorithms. We seek collective exploration algorithms with low overhead. If the explored tree was known in advance, any exploration algorithm could be viewed as centralized, since it could assume knowledge of global history by any robot at any step. However, in our case, when the topology of the tree is unknown, distributed control of robots implies that their knowledge at any step of the exploration depends on communication between them. Below we specify communication scenarios.

1.1 The model

We consider k robots initially located at the root r of an unknown tree T. Robots have distinct identifiers. Apart from that, they are identical. Each robot knows its own identifier and follows the same exploration algorithm which has the identifier as a parameter. The network is anonymous, i.e., nodes are not labeled, and ports at each node have only local labels which are distinct integers between 1 and the degree of the node. The robots move as follows. At every exploration step, every robot either traverses an edge incident to its current position, or remains in the current position. A robot traversing an edge knows local port numbers at both ends of the edge.

Our main communication scenario, called *exploration with write-read communication*, is the following. In every step of the algorithm every robot performs the following three actions: it moves to an adjacent node, writes some information in it, and then reads all information available at this node, including its degree. Alternatively, a robot can remain in the current node, in which case it skips the writing action. (This model is motivated by the capability of mobile software agents to exchange information by leaving messages in the network). Actions are assumed to be synchronous: if A is the set of robots that enter v in a given step, then first all robots from A enter v, then all robots from A write and then all robots currently located at v (those from A and those that have not moved from v in the current step) read.

We also consider two extreme communication scenarios. In one, called *exploration without communication*, all robots are oblivious of each other. I.e., at each step, every robot knows only the route it traversed until this point (which is the sequence of exit and entry port numbers), and degrees of all nodes it visited. In the other, called *exploration with complete communication*, all robots can instantly communicate at each step.

In all scenarios, a robot, currently located at a node, does not know the other endpoints of yet unexplored incident edges. If the robot decides to traverse such a new edge, the choice of the actual edge belongs to the adversary, as we are interested in the worst-case performance.

1.2 Our results

As a preliminary result, we show that the problem of finding optimal collective exploration, if the tree and the starting node are known in advance, is NP-hard. Our main result concerns collective distributed exploration of unknown trees by k robots, under the write-read communication scenario. We construct an exploration algorithm with overhead $O(k/\log k)$. Indeed, our algorithm explores any n-node tree of diameter D in time $O(D + n/\log k)$. We first describe our algorithm for the stronger scenario, exploration with complete communication, and then we show how to simulate this algorithm in the write-read model, without changing time complexity. We also prove that any algorithm must have overhead at least $2 - 1/k$ under the complete communication scenario. (This lower bound obviously carries over to the write-read communication scenario.) On the other hand we show that, in order to get overhead sublinear in the number of robots, some communication is necessary. Indeed, we prove that, under the scenario without communication, every distributed collective exploration algorithm must have overhead $\Omega(k)$. Since this is the overhead of an algorithm using only one out of k robots, our lower bound shows that exploration without communication does not allow any effective splitting of the task among robots. Comparing the upper bound on time for the scenario with write-read communication with the lower bound for the scenario without communication, shows that this difference of communication capability influences the *order of magnitude* of time of collective exploration. Even limited communication permitted by our write-read model allows robots to effectively collaborate in executing the exploration task.

1.3 Related work.

Exploration and navigation problems for robots in an unknown environment have been thoroughly investigated in recent literature (cf. the survey [23]). There are two types of models for these problems. In one of them a particular geometric setting is assumed, e.g., unknown terrain with convex obstacles [11], or room with polygonal [13] or rectangular [7] obstacles. Another approach is to model the environment as a graph, assuming that the robot may only move along its edges. The graph setting can be further specified in two different ways. In [1, 8, 9, 14] 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 [6, 12, 15–17, 21] the explored graph is undirected and the robot can traverse edges in both directions. In some papers, additional restrictions on the moves of the robot are imposed. It is assumed that the robot has either a restricted tank [6, 12], 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 [17]. It is proved in [17] that exploration can be done in time $O(e)$ under both scenarios, where e is the number of edges in the graph.

Exploration of anonymous graphs presents a different type of challenges. In this case it is impossible to explore arbitrary graphs if no marking of nodes is allowed. Hence the scenario adopted in [8, 9] was 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 concentrated attention on the minimum number of pebbles allowing efficient exploration and mapping of arbitrary directed n-node graphs. (In the case of undirected graphs, one pebble suffices for efficient exploration.) In [9] the authors compared exploration power of one robot to that of two cooperating robots with a constant number of pebbles. In [8] it was shown that one pebble is enough if the robot

knows an upper bound on the size of the graph, and $\Theta(\log \log n)$ pebbles are necessary and sufficient otherwise.

In all the above papers, except [9], exploration was performed by a single robot. Exploration by many robots was investigated mostly in the context of graphs known in advance. In [18], approximation algorithms were given for the collective exploration problem in arbitrary graphs. In [4, 5] the authors constructed approximation algorithms for the collective exploration problem in weighted trees. It was also observed in [4] that scheduling optimal collective exploration in weighted trees is NP-hard even for two robots. However, the argument from [4] does not work if all weights of edges are equal to 1, which we assume. It should also be noted that, while in [4, 5] exploration was centralized, the main focus of this paper is a distributed approach to collective tree exploration.

Another interesting study of collective exploration in unknown environments can be found, e.g., in [24, 20], in the context of a *search problem* in geometric trees and simple polygons. It should be noted that in the search problem one is interested in the detection of a single item, the *target*, as opposed to the exploration of the whole unknown environment, discussed in our paper. Moreover, the competitive ratio used in the context of geometric search is defined as the ratio between the search time and the shortest distance from the starting point to the target. Thus, in this model, an increase of the number of robots performing the search can only decrease the competitive ratio. On the other hand, in our case, where the task is to explore the whole environment, it is more appropriate to study the ratio between the time of distributed collective exploration and the optimal time of fully centralized collective exploration. In particular, as we show later, an introduction of a larger number of robots may lead, in our model, to *worse* competitive performance.

Finally, collective exploration is also related to the *freeze-tag* problem [2, 3] in which a set of "asleep" robots must be awaken, starting with only one "awake" robot. The objective is to produce an awakening schedule of minimum time. Although the task is collective in the sense that every robot participates to the process as soon as it is awaken, the freeze-tag problem has more to do with the broadcast problem than with the collective exploration problem. In particular, in the latter problem, the graph is unknown to the robots whose goal is specifically to discover this unknown environment. In contrast, the graph is given in the freeze-tag problem, and the objective is to design, in a centralized manner, a fast awakening schedule for the robots in the given graph.

2 NP-hardness of optimal collective exploration of trees

In this section, we prove a preliminary result that the problem of scheduling optimal collective exploration, if the tree and the starting node are known in advance (i.e., the problem of finding an exploration scheme working in time $opt(k, T, r)$), is NP-hard. More precisely, we consider the following optimization problem.

MIN-TIME k-ROBOTS EXPLORATION OF TREES (k-MIN-RE).
`Instance:` free tree $T = (V, E)$, $|V| = n$, $|E| = m$, a node $r \in V$, integer $k > 0$.
`Solution:` tours $C_1, \ldots, C_k$, where $\bigcup_{i=1}^{k} C_k = E$ and each tour contains node r.
`Goal:` minimize $\max\{|C_i| : i = 1, \ldots, k\}$.

In order to prove NP-hardness of k-MIN-RE, we show a transformation from the following strongly NP-complete decision problem (see [19]):

3-**PARTITION**
`Instance:` Set A of $3k$ elements, positive integer bound B, positive integer size $s(a)$ for each $a \in A$, such that $B/4 < s(a) < B/2$ and $\sum_{a \in A} s(a) = kB$.
`Question:` Can A be partitioned into k disjoint sets $A_1, \ldots, A_k$, s.t., for every $1 \leq i \leq k$, $\sum_{a \in A_i} s(a) = B$? (Where A_i must contain exactly three elements from A.)

Theorem 1. *Problem k-MIN-RE is NP-hard.*

3 Exploration with complete communication

In this section we describe and analyze an exploration algorithm for k robots, with overhead $O(k/\log k)$, under a communication model stronger than write-read communication, namely exploration with complete communication. At every step of exploration all robots exchange messages containing all information acquired to date. (Communication can be thought of as performed in a completely connected wireless network). Thus, at each step, each robot knows the part of the graph collectively explored, degrees of all visited nodes, and current positions of all robots. This strong model is introduced as an auxiliary tool, in order to explain the main idea of the algorithm and of the analysis. In the next section we show how this algorithm can be simulated in our write-read model, without changing time complexity.

We will use the following terminology. We denote by T_u the subtree of the explored tree T, rooted at node u. T_u is *explored*, if every edge of T_u has been traversed by some robot. Otherwise, it is called *unexplored*. T_u is *finished*, if it is explored and either there are no robots in it, or all robots in it are in u. Otherwise, it is called *unfinished*. T_u is *inhabited*, if there is at least one robot in it.

Algorithm Collective Exploration
Fix a step i of the algorithm and a node v in which some robots are currently located. There are three possible (exclusive) cases.
Case 1. *Subtree T_v is finished.*
`Action:` if $v \neq r$, all robots from v go to the parent of v, else all robots from v stop.
Case 2. *There exists a child u of v such that T_u is unfinished.*
Let u_1,..., u_j be children of v for which the corresponding trees are unfinished, ordered in increasing order of corresponding local port numbers at v. Let x_l be the number of robots currently located in T_{u_l}. Partition all robots from v into sets A_1,...,A_j of sizes y_1,...,y_j, respectively, so that integers $x_l + y_l$ differ by at most 1. The partition is done in such a way that indices l for which integers $x_l + y_l$ are larger by one than for some others, form an initial segment $[1, ..., z]$ in $1, ..., j$. (We will show in the proof of Lemma 1 that such a partition can be constructed). Moreover, sets A_l are formed one-by-one, by inserting robots from v in order of increasing identifiers. (Thus, the partition into sets A_1,...,A_j can be done distributedly by robots from v, using knowledge that they currently have).
`Action:` all robots from set A_l go to u_l, for $l = 1, ..., j$.
Case 3. *For all children u of v, trees T_u are finished, but at least one T_u is inhabited.*
`Action:` all robots from v remain in v.

The following lemmas will be used in the analysis of this algorithm.

Lemma 1. *Let v be any node of tree T and let i be a fixed step of Algorithm Collective Exploration. Then numbers of robots in unfinished subtrees T_u, for all children u of v, differ by at most 1.*

Proof. We prove the following assertion by induction on step i of the algorithm: "Let u_1,..., u_j be children of v for which the corresponding trees are unfinished after step i, ordered as in Case 2 of the algorithm. Then the numbers of robots in subtrees T_{u_l} differ by at most 1, and the larger numbers correspond to an initial segment in $1, ..., j$".

For $i = 1$, the assertion is obvious. Suppose that the assertion holds after step i, and consider a node v which has at least two children which are roots of unfinished subtrees after step $i + 1$. Let u_1,...,u_j be these children, ordered as in Case 2 of the algorithm. If there are no robots in v after step i then the assertion trivially holds for v after step $i + 1$, by the inductive assumption. Otherwise, the partition required in Case 2 of the algorithm is produced as follows. Suppose that there are y robots in v and x_l robots in T_{u_l}, $l = 1, ..., j$, after step i. Let $y \equiv y'$ mod j. Suppose that $x_1 = \cdots = x_m > x_{m+1} = \cdots = x_j$, where $x_m = x_{m+1} + 1$. Put $\lfloor y/j \rfloor$ robots in each set A_l. Then put one robot in each set $A_{m+1}, A_{m+2}, ..., A_j, A_1, ..., A_m$, in this order, until all robots from v are allocated. Now the numbers of robots in all trees T_{u_l}, $l = 1, ..., j$ differ by at most 1, and larger ones correspond to the initial segment $[1, ..., z]$ in $1, ..., j$, where $z = m + y'$ if $y' \leq j - m$, and $z = y' - j + m$, otherwise.

Lemma 2. *Let T_v be a subtree of tree T, and let i be the first step in which a robot enters v in the execution of Algorithm Collective Exploration. If T_v has m edges then T_v is finished by step $i + 2m$.*

Proof. The proof is by induction on the height of T_v. If v is a leaf, the lemma is obvious. Otherwise, fix any robot R which enters v in step i. Let u_1,..., u_j be those children of v which R visits, in the order of visits. Suppose that T_{u_l} has m_l edges. By the inductive hypothesis, the lemma is true for all T_{u_l}. Hence, by step $i + \sum_{l=1}^{j}(2 + 2m_l) \leq i + 2m$, all subtrees T_{u_l} will be finished, and R will be back at v. If u_1,..., u_j are the only children of v then T_v is already finished. If not, then $i + \sum_{l=1}^{j}(2 + 2m_l) < i + 2m$. In this case, all other children u of v must be finished by step $i + \sum_{l=1}^{j}(2 + 2m_l)$, otherwise R would visit one of them in the next step. Hence T_v is finished by step $i + \sum_{l=1}^{j}(2 + 2m_l) + 1 \leq i + 2m$.

Lemma 3. *Algorithm Collective Exploration works in time $O(D + n/\log k)$ for all n-node trees of diameter D.*

Proof. Consider Algorithm Collective Exploration, working on a tree T of diameter D, rooted at r. Define a path $S = (a_0, a_1, ...)$ in T as follows. $a_0 = r$. Suppose that a_j is already defined. Among all children of a_j consider those nodes v for which T_v was finished last (there can be several such children). Define a_{j+1} to be such a child with smallest port label. The length $|S|$ of S is at most D. Intuitively, the path S leads to one of the leaves explored very late.

For any positive integer i and for any $j = 0, \ldots, \log k$, denote by $p_i(j)$ the largest index of a node v on path S such that there are at least 2^j robots in T_v after step i. We will say that $p_i(j)$ *corresponds* to the node with this index. Define nodes $w_i(l)$, for $|S| \geq l \geq 1$, as follows. Let $w_i(l)$ denote the lth node on S which has at least two children u_1 and u_2, such that T_{u_1} and T_{u_2} are inhabited after step i. Let $d_i(l)$, for $l \geq 1$, denote the number of such children of node $w_i(l)$.

Define i_0 to be the last step of the algorithm satisfying the following condition: for all $i \leq i_0$, $p_i(0)$ is smaller than the length of S. We first consider only steps of the algorithm until step i_0. We define two types of such steps. A step $i \leq i_0$ of the algorithm is of type

A. if $\sum_l d_i(l) \geq \frac{1}{2}\log k$;
B. if $|\{j : p_{i+1}(j) \neq p_i(j)\}| \geq \frac{1}{4}(\log k + 1)$.

We now show that all steps of the algorithm are of one of the above types. The proof of this fact is split into the following three claims.

Claim 1. *Fix a step $i \leq i_0$ of the algorithm, and consider a node $w_i(l)$, for some $l \geq 1$. Then $|\{j : p_i(j)$ corresponds to node $w_i(l)\}| \leq d_i(l) + 1$.*

Let v denote the successor of $w_i(l)$ on path S (v exists by definition of i_0). Let j_0 be the smallest element in the set $\{j : p_i(j)$ corresponds to node $w_i(l)\}$. The number of robots in T_v, after step i, is $x < 2^{j_0}$. By the definition of $d_i(l)$ and by Lemma 1, the number of robots in $T_{w_i(l)}$ is less than $(x+1) \cdot d_i(l)$. We have

$$\begin{aligned}(x+1) \cdot d_i(l) &\leq x \cdot d_i(l) + d_i(l) \\ &\leq x \cdot 2^{d_i(l)} + 2^{d_i(l)} \\ &\leq x \cdot 2^{d_i(l)} + x \cdot 2^{d_i(l)} \\ &= x \cdot 2^{d_i(l)+1} \\ &< 2^{j_0 + d_i(l) + 1}.\end{aligned}$$

Hence, if $p_i(j)$ corresponds to $w_i(l)$ then $j < j_0 + d_i(l) + 1$. This proves Claim 1.

Claim 2. *Fix a step $i \leq i_0$ of the algorithm. If $p_i(j)$ does not correspond to any $w_i(l)$, for $l \geq 1$, then $p_{i+1}(j) \neq p_i(j)$.*

Consider $p_i(j)$ satisfying the assumption of Claim 2. Let v denote the corresponding node on path S, and let v' denote the successor of v on S. The number of robots in T_v is equal to the number of robots in v plus the number of robots in $T_{v'}$, in view of the fact that $p_i(j)$ does not correspond to any $w_i(l)$ and of the definition of $w_i(l)$. In step $i+1$, all robots from v move to v', and all robots located in $T_{v'}$ remain in $T_{v'}$. (Indeed, since $i \leq i_0$, $T_{v'}$ has not yet been explored, hence it has not been finished, and all subtrees rooted at siblings of v' are finished and not inhabited, by the assumption that $p_i(j)$ does not correspond to any $w_i(l)$.) Hence $p_{i+1}(j)$ corresponds to v'. This proves Claim 2.

Claim 3. *All steps $i \leq i_0$ of the algorithm are either of type A or of type B.*

Fix a step $i \leq i_0$ of the algorithm and suppose that it is not of type A. Hence $\sum_l d_i(l) < \frac{1}{2}\log k$. Since $d_i(l) \geq 2$, for all $l \geq 1$, the number of indices l for which $d_i(l)$ are defined, is less than $\frac{1}{4}\log k$. It follows that $\sum_l (d_i(l)+1) < \frac{1}{2}\log k + \frac{1}{4}\log k = \frac{3}{4}\log k$. By Claim 1, the number of integers j, such that $p_i(j)$ does not correspond to any $w_i(l)$, for $l \geq 1$, is larger than $\log k + 1 - \frac{3}{4}\log k > \frac{1}{4}(\log k + 1)$. By Claim 2, the number of integers j, such that $p_{i+1}(j) \neq p_i(j)$, is also larger than $\frac{1}{4}(\log k + 1)$. Hence step i is of type B. This proves Claim 3.

We now estimate the number of steps of type A. Consider all subtrees T_u rooted at nodes u outside of S. Let x_u denote the number of edges of T_u. We have $\sum_u (x_u + 1) \leq n$. Let t_u denote the number of steps during which T_u is inhabited. By Lemma 2, $\sum_u t_u \leq 2n$. In every step i of type A, at least $\sum_l (d_i(l) - 1)$ trees T_u are inhabited (subtrees T_u are rooted at nodes u outside of S, hence summands are $d_i(l) - 1$). Since $d_i(l) \geq 2$, we have $\sum_l (d_i(l) - 1) \geq (\sum_l d_i(l))/2 \geq \frac{1}{4}\log k$. Hence the number of steps of type A is at most $\frac{2n}{\frac{1}{4}\log k} = \frac{8n}{\log k}$.

Next, we estimate the number of steps of type B. We have

$$\sum_{i \leq i_0} |\{j : p_{i+1}(j) \neq p_i(j)\}| = |\bigcup_{i \leq i_0} \bigcup_{j=0}^{\log k} \{(i,j) : p_{i+1}(j) \neq p_i(j)\}| =$$

$$|\bigcup_{j=0}^{\log k}\bigcup_{i\leq i_0}\{(i,j): p_{i+1}(j)\neq p_i(j)\}| = \sum_{j=0}^{\log k}|\{i: p_{i+1}(j)\neq p_i(j)\}| \leq (\log k+1)\cdot|S|,$$

the last inequality following from the fact that before step i_0 all moves of robots on S are down the path S, and hence, for a given j, the size of the set $\{(i,j): p_{i+1}(j)\neq p_i(j)\}$ is bounded by the length of S. For every step i of type B, we have $|\{j: p_{i+1}(j)\neq p_i(j)\}|\geq\frac{1}{4}(\log k+1)$, hence the number of steps of type B is at most $\frac{|S|(\log k+1)}{\frac{1}{4}(\log k+1)}=4|S|$. Hence, by Claim 3, we have $i_0\leq\frac{8n}{\log k}+4|S|$.

We finally show that the algorithm completes exploration by step $i_0+1+|S|$. Let $i_1=i_0+1$. Let X be the set of robots that are in the last node b of S after step i_1. In step i_1+1, all robots from X go to the parent of b, because b is a leaf. By definition of S, when a set of robots containing X moves from a node v' on S to its parent v, then $T_{v'}$ is finished and not inhabited, and consequently, by the construction of v', T_v is also finished. It follows that in the next step, all robots from v move to the parent of v. Hence the number of steps after i_1, needed to terminate the algorithm, is $|S|$. This implies that the algorithm terminates by step $i_1+|S|=i_0+1+|S|$. Hence the running time of the algorithm is at most $\frac{8n}{\log k}+5|S|+1\in O(D+n/\log k)$.

Theorem 2. *Algorithm Collective Exploration has overhead $O(k/\log k)$.*

Proof. Consider any n-node tree T rooted at node r. If the diameter of T is at most $\frac{n\log k}{k}$ then the theorem follows from Lemma 3, because $opt(k,T,r)\geq 2(n-1)/k$. If the diameter of T is larger than $\frac{n\log k}{k}$ then $opt(k,T,r)\in\Omega(\frac{n\log k}{k})$, because at least one robot has to visit the leaf farthest from r. By Lemma 2, Algorithm Collective Exploration uses time $\leq 2n$, hence the overhead is $O(k/\log k)$ in this case as well.

We conclude this section by stating a lower bound on the overhead of any collective exploration under the complete communication scenario. Clearly, this lower bound also holds under the write-read communication scenario. The proof is omitted.

Theorem 3. *Any collective exploration algorithm for k robots has overhead $\geq 2-1/k$.*

4 Exploration with write-read communication

In this section we show how Algorithm Collective Exploration can be simulated in our write-read model, without changing time complexity. Fix any node v of the tree. Let i denote the step number, and let p denote the port number at v corresponding to the parent of v; in the case $v=r$, we define $p=*$. We define the following sets:

- $\mathcal{P}_i$ is the set of ports at v corresponding to children which are roots of unfinished subtrees,
- $\mathcal{P}'_i\subseteq\mathcal{P}_i$ is the set of ports at v corresponding to children in whose subtrees there is one robot more than in subtrees of all other children. In the special case when all subtrees of children are inhabited by q robots, we define $\mathcal{P}'_i=\mathcal{P}_i$, if $q>0$, and $\mathcal{P}'_i=\emptyset$, if $q=0$.
- $\mathcal{R}_i$ is the set of identifiers of robots that are in v after step $i-1$.

Let $\mathcal{K}_i=\{p,\mathcal{P}_i,\mathcal{P}'_i,\mathcal{R}_i\}$, if node v has been visited by step $i-1$ of Algorithm Collective Exploration. Otherwise $\mathcal{K}_i$ is undefined. We refer to $\mathcal{K}_i$ as the knowledge at node v after step $i-1$ of Algorithm Collective Exploration. The action performed by every robot located at v after step $i-1$ depends only on $\mathcal{K}_i$ and on the identifier of the

robot. Hence Algorithm Collective Exploration defines the following *action function* H. For any step i and any robot R located at v after step $i-1$, the value of $H(\mathcal{K}_i, R)$ is one of the following:

- the port number α by which R leaves v in step i,
- 0, if R remains at v in step i,
- $*$, if R stops.

We construct a simulation of Algorithm Collective Exploration in the write-read communication model. The new algorithm is called Algorithm Write-Read. It operates in *rounds* logically corresponding to steps of Algorithm Collective Exploration. Each round $i > 0$ consists of three steps, $3i$, $3i+1$, $3i+2$, and round 0 consists of two steps, 1 and 2. Each step is in turn divided into three stages: in Stage 1 robots move, in Stage 2 they write information in their location, and in Stage 3 they read information previously written in their location.

Recall that, in the write-read model, any robot R entering node v can write some information in this node. In the Algorithm Write-Read, a robot R entering node v in step i using port α, writes the triplet (i, R, α) at node v. Denote by $\mathcal{I}_i$ the set consisting of the degree of v and of all triplets written at node v until step $i-1$ of Algorithm Write-Read.

We now define the knowledge $\hat{\mathcal{K}}_i$ at v after round $i-1$ of Algorithm Write-Read. If no triplets are written at node v then $\hat{\mathcal{K}}_i$ is not defined. Otherwise, we define $\hat{\mathcal{K}}_i = \{p, \hat{\mathcal{P}}_i, \hat{\mathcal{P}}'_i, \hat{\mathcal{R}}_i\}$, where $\hat{\mathcal{P}}_i$, $\hat{\mathcal{P}}'_i$, $\hat{\mathcal{R}}_i$ are defined with respect to Algorithm Write-Read (after round $i-1$) in the same way as $\mathcal{P}_i$, $\mathcal{P}'_i$, $\mathcal{R}_i$ were defined with respect to Algorithm Collective Exploration (after step $i-1$). We will show that, at the beginning of each round i of Algorithm Write-Read, any robot located at v knows $\hat{\mathcal{K}}_i$. Moreover, we will show that, for any v and any i, $\hat{\mathcal{K}}_i = \mathcal{K}_i$, and that $\hat{\mathcal{K}}_i$ is defined for exactly the same nodes as $\mathcal{K}_i$.

Knowledge $\hat{\mathcal{K}}_i$ is obtained from input $\mathcal{I}_{3i}$ by the following recursive procedure.

Procedure Knowledge Construction

Assume that knowledge $\hat{\mathcal{K}}_1$ is undefined at nodes other than r and that it equals to $\{*, \hat{\mathcal{P}}_1, \hat{\mathcal{P}}'_1, \hat{\mathcal{R}}_1\}$ at the root r, where $\hat{\mathcal{P}}_1$ is the set of all ports of r, $\hat{\mathcal{P}}'_1 = \emptyset$, and $\hat{\mathcal{R}}_1$ is the set of all robots. Suppose that we can compute $\hat{\mathcal{K}}_i$ from input $\mathcal{I}_{3i}$, at all nodes v. We show how to compute $\hat{\mathcal{K}}_{i+1}$ from $\mathcal{I}_{3i+3}$, at node v.

(1) If there are no triplets written at node v for steps smaller than $3i$ (i.e., $\hat{\mathcal{K}}_i$ is undefined) but there is some triplet $(3i, R, \alpha) \in \mathcal{I}_{3i+3}$ then we put: $p = \alpha$ (there is exactly one such α in this case), $\hat{\mathcal{P}}_{i+1}$ is the set of all ports at v other than α, $\hat{\mathcal{P}}'_{i+1} = \emptyset$, $\hat{\mathcal{R}}_{i+1}$ is the set of all robots R, such that a triplet $(3i, R, \alpha) \in \mathcal{I}_{3i+3}$ is written in v.

(2) Otherwise, we first put $\hat{\mathcal{K}}_{i+1} = \hat{\mathcal{K}}_i$, and then modify $\hat{\mathcal{K}}_{i+1}$, s.t.: p remains unchanged, $\hat{\mathcal{P}}_{i+1}$ is the set of all ports from $\hat{\mathcal{P}}_i$, except those ports α, for which there is a triplet $(3i+1, R, \alpha) \in \mathcal{I}_{3i+3}$ at v (we discard those ports by which a robot entered confirming that the corresponding subtree is finished), $\hat{\mathcal{P}}'_{i+1}$ contains z initial ports from $\hat{\mathcal{P}}_{i+1}$, where z is the integer defined in step i of Algorithm Collective Exploration, $\hat{\mathcal{R}}_{i+1} := \hat{\mathcal{R}}_i \cup X \setminus Y$, where X is the set of robots R for which $(3i, R, \alpha) \in \mathcal{I}_{3i+3}$, and Y is the set of robots R' for which $H(\mathcal{K}_i, R') = \alpha \neq 0$ (we add robots that entered v in step $3i$ and delete those that left v in this step).

Algorithm Write-Read

Round 0 - This is a special round used to distinguish the root r.

- - STEP 1: - -

stage 1: do nothing.

stage 2: every robot R writes $(1, R, *)$ at node r.

stage 3: every robot R reads $\mathcal{I}_1$ at node r.

- - STEP 2: - - (do nothing).

Round $i > 0$ - Execution of each round is based on two assumptions

The assumptions after round $i-1$ are: assumption A_i - $\hat{\mathcal{K}}_i$ is correctly computed by Procedure Knowledge Construction, using $\mathcal{I}_{3i}$; and assumption B_i - $\hat{\mathcal{K}}_i = \mathcal{K}_i$, at any node, and $\hat{\mathcal{K}}_i$ is defined for exactly the same nodes as $\mathcal{K}_i$.

The three steps of each round i have the following purpose. Step $3i$ is used to make the actual move of a robot to its new location, according to the simulated Algorithm Collective Exploration. Step $3i+1$ is used to temporarily move robots from a node whose subtree is finished, to its parent w, in order to update information held at w, concerning children with finished subtrees. Step $3i+2$ is used to move back robots that temporarily moved in Step $3i+1$.

- - STEP $3i$: - -

stage 1: If R is at node r at the end of round $i-1$, and $H(\mathcal{K}_i, R) = *$ for node r, then R stops. If R is at node v at the end of round $i-1$, and $H(\mathcal{K}_i, R) = \alpha \notin \{0, *\}$, for node v, then R leaves v through port α.

stage 2: Every robot R that entered v through port α in Stage 1 of Step $3i$, writes $(3i, R, \alpha)$ in node v.

stage 3: Every robot R located at v reads $\mathcal{I}_{3i+1}$ (this is information held at v after Stage 2 of Step $3i$.)

- - STEP $3i+1$: - -

stage 1: If $\mathcal{P}_i = \emptyset$ then every robot R located at v at the end of Step $3i$ leaves v through port p.

stage 2: Every robot R that entered v through port α in Stage 1 of Step $3i+1$, writes $(3i+1, R, \alpha)$ at node v.

stage 3: Every robot R located at v reads $\mathcal{I}_{3i+2}$.

- - STEP $3i+2$: - -

stage 1: Every robot R that entered v through port α in Step $3i+1$, leaves v through port α.

stage 2: Every robot R that entered v through port α in Stage 1 of Step $3i+2$, writes $(3i+2, R, \alpha)$ in node v.

stage 3: Every robot R located at v reads $\mathcal{I}_{3i+3}$.

Remark. The return moves of robots in stage 1 of step $3i+2$ could be avoided. They are introduced to simplify analysis of knowledge update, and do not influence exploration complexity.

Lemma 4. *Assumptions A_i & B_i from Algorithm Write-Read are satisfied for all $i > 0$.*

Theorem 4. *Algorithm Write-Read works in time $O(D + \frac{n}{\log k})$ for all n-node trees of diameter D.*

Proof. By Lemma 3, it is enough to show that, for every tree T rooted at r, the number of rounds used by Algorithm Write-Read is not larger than the number of steps used by Algorithm Collective Exploration. Let i_0 denote the latter number. By Lemma 4, assumptions A_{i_0} and B_{i_0} are satisfied. By assumption B_{i_0}, all robots are at the root

r after round $i_0 - 1$, because they are all at the root after step $i_0 - 1$ of Algorithm Collective Exploration. In Step $3i_0$ of Algorithm Write-Read, every robot R performs action $H(\hat{\mathcal{K}}_{i_0}, R)$, by assumption A_{i_0}. This action is equal $H(\mathcal{K}_{i_0}, R)$, by assumption B_{i_0}. By the definition of i_0 this action is stop. Hence all robots stop after round i_0 of Algorithm Write-Read.

Corollary 1. *Algorithm Write-Read has overhead $O(k/\log k)$.*

5 Exploration without communication

In this section we show that, in the absence of communication between robots, the overhead of any exploration algorithm is $\Omega(k)$, i.e., of the same order of magnitude as if only one out of k robots were used to explore the tree. This shows that without communication between robots, no effective advantage can be taken of collective exploration.

Theorem 5. *Every collective exploration algorithm for k robots, under the scenario without communication, has overhead $\Omega(k)$.*

Proof. Let $\mathcal{A}$ be any exploration algorithm for k robots, under the scenario without communication. We show that, for any k and $n = k^2 + k$, there exists an n-node tree T and a node r of T, such that $\mathcal{A}(k, T, r)/opt(k, T, r) \in \Omega(k)$.

The tree T is of size $n = k^2 + k$, has root r, and $k + 1$ levels: level 0 consists of the root, each of levels 1,..., $k - 1$, has size $k + 1$, and level k has size k. All nodes at level $i + 1$ have the same parent at level i. Call this parent the main node at level i. We say that a robot is delayed at level i, if it spends there at least time k before finding the main node. Consider levels 1,...,m, where $m = k - 1$.

Fix a robot R, and pick the main node at each level randomly, with uniform probability. Hence R is delayed at level i with probability at least $1/2$. Consequently, the probability that R is delayed at fewer than $m/4$ of the first m levels, is at most $2^{-\Omega(k)}$. (The expected value of the number of levels at which R is delayed is $m/2$, hence the estimate is obtained by Chernoff's bound). Hence the probability that some robot is delayed at fewer than $m/4$ of the first m levels is at most $k \cdot 2^{-\Omega(k)} < 1$, for sufficiently large k. Hence there is a choice of main nodes, such that all robots are delayed at least $m/4$ of the first m levels. This means that every robot arrives at level $k = m + 1$ after time at least $(k - m/4) + km/4 \in k^2/4 - O(k)$.

On the other hand, if the tree T is known, then robots can use time $k - 1$ to position themselves evenly at levels 0, 1, ..., $k - 1$ on the main branch, and then complete the task in time $2k$, in parallel for all levels. Then all of them must get back to the root. This gives time $2k + 2(k - 1)$. The ratio is $k/16 - O(1)$.

6 Conclusion

We showed that collective tree exploration can be done faster, if robots have some communication capabilities. This result should be considered a first step in the study of the impact of communication between robots on the efficiency of collective network exploration. Several related problems remain open, including: (1) find a tree exploration algorithm with constant overhead in the complete communication scenario; (2) find a good lower bound on the overhead of tree exploration for the write-read model; (3) generalize our results to exploration of arbitrary networks; and (4) consider other communication models in the context of collective network exploration.

References

1. S. Albers and M. R. Henzinger, Exploring unknown environments, *SIAM Journal on Computing*, **29** (2000), 1164-1188.
2. E. Arkin, M. Bender, S. Fekete, J. Mitchell, and M. Skutella, The freeze-tag problem: How to wake up a swarm of robots, In *13th ACM-SIAM Symp. on Disc. Alg.* (SODA'02), 568-577.
3. E. Arkin, M. Bender, D. Ge, S. He, and J. Mitchell. Improved approximation algorithms for the freeze-tag problem. In *15th ACM Symp. on Par. in Alg. and Arch.* (SPAA'03), 295-303.
4. I. Averbakh and O. Berman, A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree, *Discr. Appl. Mathematics*, **68**, (1996), 17-32.
5. I. Averbakh and O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, *Discr. Appl. Mathematics*, **75**, (1997), 201-216.
6. B. Awerbuch, M. Betke, R. Rivest and M. Singh, Piecemeal graph learning by a mobile robot, In *8th Conf. on Comput. Learning Theory* (COLT'95), 321-328.
7. E. Bar-Eli, P. Berman, A. Fiat and R. Yan, On-line navigation in a room, *Journal of Algorithms*, **17**, (1994), 319-341.
8. M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs, In *30th Ann. Symp. on Theory of Comp.* (STOC'98), 269-278.
9. M.A. Bender and D. Slonim, The power of team exploration: Two robots can learn unlabeled directed graphs, In *35th Ann. Symp. on Foundations of Comp. Science* (FOCS'96), 75-85.
10. P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen and M. Saks, Randomized robot navigation algorithms, In *7th ACM-SIAM Symp. on Discrete Algorithms*, (SODA'96), 74-84.
11. A. Blum, P. Raghavan and B. Schieber, Navigating in unfamiliar geometric terrain, *SIAM Journal on Computing*, **26**, (1997), 110-137.
12. M. Betke, R. Rivest and M. Singh, Piecemeal learning of an unknown environment, *Machine Learning*, **18**, (1995), 231-254.
13. X. Deng, T. Kameda and C. H. Papadimitriou, How to learn an unknown environment I: the rectilinear case, *Journal of the ACM*, **45**, (1998), 215-245.
14. X. Deng and C. H. Papadimitriou, Exploring an unknown graph, *J. of Graph Th.*, **32**, (1999), 265-297.
15. A. Dessmark and A. Pelc, Optimal graph exploration without good maps, In *10th European Symposium on Algorithms* (ESA'02), 374-386.
16. K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc, Tree exploration with little memory, In *13th Ann. ACM-SIAM Symposium on Discrete Algorithms* (SODA'02), 588-597.
17. C.A. Duncan, S.G. Kobourov and V.S.A. Kumar, Optimal constrained graph exploration, In *12th Ann. ACM-SIAM Symp. on Discrete Algorithms*, (SODA'01), 807-814.
18. G. N. Frederickson, M. S. Hecht and C. E. Kim, Approximation algorithms for some routing problems, *SIAM J. on Computing*, **7**, (1978), 178-193.
19. M. Garey, and D. Johnson, Computers and Intractability, W.H. Freeman and Company, New York, 1979.
20. A. Lopez-Ortiz and S. Schuierer, On-line Parallel Heuristics and Robot Searching under the Competitive Framework. In *8th Scandinavian Work. on Alg. Theory*, (SWAT'02), 260-269.
21. P. Panaite and A. Pelc, Exploring unknown undirected graphs, *J. of Algorithms*, **33**, (1999), 281-295.
22. C. H. Papadimitriou and M. Yannakakis, Shortest paths without a map, *Theoretical Computer Science*, **84**, (1991), 127-150.
23. N. S. V. Rao, S. Hareti, W. Shi and S.S. Iyengar, Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, *Tech. Rep. ORNL/TM-12410*, Oak Ridge National Laboratory, July 1993.
24. S. Schuierer, On-line searching in geometric trees, In *Sensor Based Intelligent Robots*, LNAI 1724, 1999, 220-239.