A Recursive Approach to Multi-Robot Exploration of Trees

Christian Ortolf and Christian Schindelhauer

University of Freiburg, Department of Computer Science, Computer Networks
{ortolf,schindel}@informatik.uni-freiburg.de

Abstract. The multi-robot exploration problem is to explore an unknown graph of size n and depth d with k robots starting from the same node. For known graphs a traversal of all nodes takes at most $\mathcal{O}(d + n/k)$ steps. The ratio between the time until cooperating robots explore an unknown graph and the optimal traversal of a known graph is called the competitive exploration time ratio.
It is known that for any algorithm this ratio is at least $\Omega\left((\log k)/\log\log k\right)$. For $k \leq n$ robots the best algorithm known so far achieves a competitive time ratio of $\mathcal{O}\left(k/\log k\right)$.
Here, we improve this bound for trees with bounded depth or a minimum number of robots. Starting from a simple $\mathcal{O}(d)$-competitive algorithm, called Yo-yo, we recursively improve it by the Yo-star algorithm, which for any $0 < \alpha < 1$ transforms a $g(d,k)$-competitive algorithm into a $\mathcal{O}((g(d^\alpha,k)\log k + d^{1-\alpha})(\log k + \log n))$-competitive algorithm. So, we achieve a competitive bound of $\mathcal{O}\left(2^{\mathcal{O}(\sqrt{(\log d)(\log\log k)})}(\log k)(\log k + \log n)\right)$. This improves the best known bounds for trees of depth d, whenever the number of robots is at least $k = 2^{\omega(\sqrt{(\log d)(\log\log d)})}$ and $n = 2^{O(2^{\sqrt{\log d}})}$.

Keywords: competitive analysis, robot, collective graph exploration

1 Introduction

Maintenance robots are nowadays common for households and every day life. One of the most basic tasks such robotic lawn mowers, vacuum cleaners, and underwater cleaning robots is to explore a new environment. Besides the technical problem of localization, orientation and communication it is not clear how to make use of the full potential of the parallel exploration. This is the problem setting of multi-robot exploration.

The Model. The multi-robot exploration takes place on a labeled, connected, undirected graph $G = (V, E)$ with $|V| = n$ and diameter d. In each round an algorithm has to decide for each of the k robots which edge it traverses to a neighboring node. The algorithm knows the positions of all robots and is not computationally restricted. The goal is to visit all nodes of the graph as fast as possible at least once.

An offline algorithm has full knowledge of G, while an online algorithm can only use the induced subgraph defined by the already visited nodes of V and their neighboring nodes. The efficiency of an online algorithm is measured by comparing its run-time

against the asymptotically optimal offline algorithm. The maximum of the ratio between the run-time of the online and the offline exploration time is called the competitive ratio of an algorithm.

Our Results In this paper we present two algorithms. The first, called Yo-yo, achieves competitive ratio of $4d$, while the second, called Yo*, recursively improves this bound up to a ratio of $\mathcal{O}\left(2^{\mathcal{O}(\sqrt{(\log d)(\log\log k)})}(\log k)(\log k + \log n)\right)$. This is an improvement of the best known results for a minimum number of $k = d^c$ robots for $c > 0$ up to a ratio of $k^\epsilon \log n$ for any $\epsilon > 0$.

Related work Exploration is a more than a century old problem (for a survey we recommend [18]) closely related to the Traveling Salesman and the Hamiltonian cycle problem. But most work on exploration only handles various cases concerning a single robot.

For the single robot case asymptotically optimal exploration up to a factor of two is possible with depth-first-search DFS. Using a map the exploration of a line or tree can be improved by preventing double traversal of edges. Desmark et al. show in [8] various competitive constants that can be gained depending on if an anchored, unanchored or no map at all is available.

If graphs are not labeled, DFS cannot be directly used. M.A. Bender presents solutions for this scenario in [3, 4] using a pebble or a second robot for bookkeeping.

In 2006, Fraigniaud et al. consider the multi-robot exploration problem for trees in [12]. They present an algorithm that with a run-time of $\mathcal{O}(k/\log k)$ is far apart from their lower bound of $\Omega(2+1/k)$ and quite close the trivial upper bound of $\mathcal{O}(k)$ achieved by executing a depth first search using a single robot. While the lower bound is improved by Dynia et al. in [10] to $\Omega(\frac{\log k}{\log\log k})$ the upper bound still is the state of the art and the exponential gap remains open between these two bounds.

Several restrictions for the exploration can improve the bounds. If algorithms are restricted to greedy exploration an even stronger bound of $\Omega(k/\log k)$ is shown by Higashikawa et al. [15].

For restricted graphs several better algorithms exist. Dynia et al. showed in [9] a faster exploration for trees restricted by a *density* parameter p, enforcing a minimum depth for any subtree depending on its size. For example trees embeddable in p-dimensional grids could be explored with competitiveness of $\mathcal{O}(d^{1-1/p})$.

For 2-dimensional grids with only convex obstacles we improved the competitive bound to $\mathcal{O}(\log^2 n)$ in [17]. Also note that despite this strong restrictions to a graph the same lower bound of $\Omega(\frac{\log k}{\log\log k})$ as for trees holds.

In Brass et al.'s work an upper bound of $\mathcal{O}(\frac{n}{k} + d^{k-1})$ [5] is shown, they implement an algorithm that moves robots similar to the method of Fraigniaud et al. [12], but also works on graphs using only a local communication model with bookkeeping devices.

Dereniowski et al. discuss in their work very large values of k. They show how many more robots need to be invested to explore in asymptotically optimal time. They show a minimum of $k = dn^{1+\epsilon}$ for an $\epsilon > 0$ robots to be necessary and improve with this the trivial bound of $\mathcal{O}(n^d)$ required to explore any graph in time d with flooding [7].

Exploration of directed graphs is not discussed here. Competitive analysis done by Albers et al. [1], Fleischer et al. [11], Papadimitriou et al. [6] and Förster et al. [13] indicates this to be a harder problem than the undirected case.

Some works model the exploration geometrically, this is useful if robots have a sense of sight enabling to see additional nodes before visiting them [14, 16] or having to move around corners to make everything visible in case of unlimited vision [2].

A constant factor offline approximation. A very basic observation is the constant factor offline approximation presented in Algorithm 1, which establishes a constant approximation factor of four.

Algorithm 1: Offline 4-competitive multi-robot exploration of trees for robot j

1: Compute a cycle of length $2n$ using DFS covering the tree
2: Divide the cycle into k intervals of size at most $\lceil 2n/k \rceil$
3: Go to the j-th interval
4: Traverse the interval

Lemma 1. *Algorithm 1 needs at most $d + \lceil 2n/k \rceil$ robot moves and has a competitive factor of four.*

Proof. Every exploration algorithm needs at least $\max\{\lceil n/k \rceil, d\}$ steps. The number of robot moves of Algorithm 1 is $d + \lceil 2n/k \rceil \leq 2\max\{d, \lceil 2n/k \rceil\} \leq 4\max\{d, \lceil n/k \rceil\}$. Hence, it is 4-competitive.

2 The Yo-yo Exploration

The basic idea of the Yo-yo exploration algorithm is to successively explore every set of nodes in the tree with the same depth. After each exploration step all robots return to the root and are perfectly rebalanced for the next exploration step. For most trees this algorithm is not very efficient, since most of the time the robots commute between the root and the leafs of the so far known sub-tree. We denote the number of nodes in depth i by n_i.

Algorithm 2: The Yo-yo Algorithm: $4d$-competitive multi-robot exploration of a tree

1: All robots start at the root of the tree
2: **for** $i \leftarrow 2, \ldots, d$ **do**
3: Partition all n_i nodes in depth i into k subsets $V_{i,1}, \ldots, V_{i,k}$ with $|V_{i,j}| \leq \lceil \frac{n_i}{k} \rceil$.
4: **for all** $j \leftarrow 1, \ldots, k$ **do in parallel**
5: **for all** $u \in V_{i,j}$ **do**
6: Move robot j to u
7: Move robot j to the root
8: **end for**
9: **end for**
10: **end for**

The main motivation of this algorithm is that the competitive ratio $4d$ only depends on the depth d, which we can improve later on by a technique which does not work with a competitive factor only depending on k. Note that for at least $k = \Omega(d \log d)$ robots Yo-yo is asymptotically at least as good as the best known algorithm of Fraigniaud et al. [12].

Lemma 2. *The Yo-yo algorithm needs at most $d(d+1) + 2dn/k$ rounds to explore a graph with n nodes, depth d and k robots, and thus has a competitive exploration ratio of at most $4d$.*

Proof. The success of the exploration algorithm follows by an easy induction over the tree depth. For the number of rounds, note that in lines 6 and 7 each of the k robots moves for $2i$ rounds in order to explore a node in $V_{i,j}$ and return to the root. This is repeated in the loop starting at line 5 for at most $\lceil n_i/k \rceil$ times. Therefore, the overall number of rounds for the outer loop starting at line 2 is the following.

$$\begin{aligned}\sum_{i=1}^{d} 2i \left\lceil \frac{n_i}{k} \right\rceil &\leq \sum_{i=1}^{d} 2i \left(1 + \frac{n_i}{k}\right) \\ &= d(d+1) + \sum_{i=1}^{d} 2i \frac{n_i}{k} \\ &\leq d(d+1) + 2d\frac{n}{k}\,,\end{aligned}$$

where we use $\sum_{i=0}^{d} n_i = n$. Now, every exploration algorithm needs at least $\max\{d, \lceil n/k \rceil\} \geq \frac{1}{2}(d + n/k)$ rounds. So, the competetive factor is at most

$$\frac{d(d+1) + 2d\frac{n}{k}}{\max\{d, \lceil n/k \rceil\}} \leq \frac{2d^2 + 2d\frac{n}{k}}{\frac{1}{2}(d + n/k)} \leq 4d$$

□

3 The Yo* Algorithm

Starting from the Yo-yo algorithm (Algorithm 3) we use a recursive approach to improve the efficiency of the exploration. To avoid the rebalancing step passing the root in each step we divide the graph into the uppermost segment of depth c and b segments of depth a such that $d = ab + c$ which values are to be chosen later on, see Fig. 1. The first segment will be explored by the base algorithm, e.g. the Yo-yo algorithm. One can easily see that if the competitive ratio grows with the depth of the tree we can bound the ratio with a smaller term now.

All deeper unexplored segments will be handled together with the last explored segment,see Fig. 2. These two segments form a forest of trees. If the number of trees is greater than the number of robots, we can use DFS to efficiently explore them. However, the size of the trees can differ and therefore, we rebalance the robots if half of the trees

have been explored by DFS. The rebalancing costs at most d steps and this has to be repeated at most $\log n$ times.

If the number of trees has been reduced to be smaller than the number of robots, we use the base algorithm and rebalance again, if half of the trees have been completely explored. So, we have $\log k$ iterations for all the b segments.

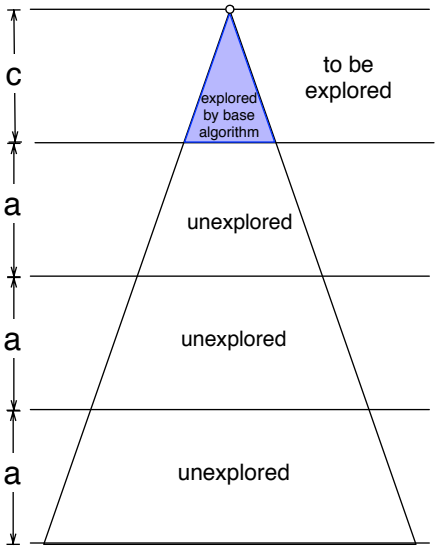

Fig. 1. The first round of the Yo-star algorithm

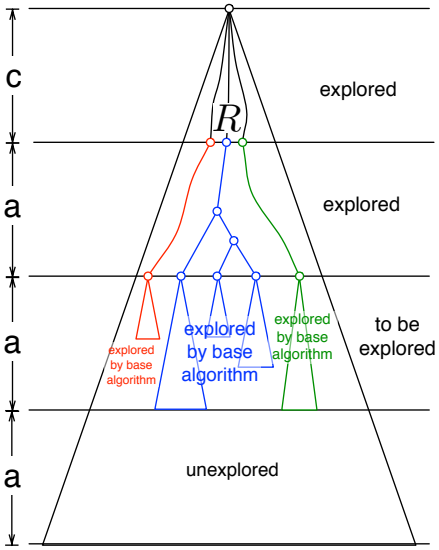

Fig. 2. The principle of the Yo-star algorithm

Taking the Yo-yo algorithm and choosing segments of depth $a = c = d^{\frac{1}{2}}$ a back-on-the-envelope calculation gives us a competitive ratio of $O(d^{\frac{1}{2}})$ for the first segment and a ratio of $O(d^{\frac{1}{2}}(\log n + \log k))$ for all the other segments. So, after one iteration of the Yo* algorithm we improve the depth-dependent factor in the ratio from d to $d^{\frac{1}{2}}$. Now, if we take this new algorithm as base algorithm and choose segments of size $d^{\frac{1}{3}}$ we improve the ratio to $d^{\frac{1}{3}}$. However, there is an overhead in the iteration, where constant factors grow exponentially over the number of iterations and thus must be carefully analyzed.

So, we assume we start from a $g(d,k)(d+\frac{n}{k})$ time bounded algorithm and try to turn it into a more efficient one using the Yo* algorithm. From $g(d,k)$ we only know that it is a monotone increasing function with respect to d and k, e.g. for the Yo-yo algorithm we have $g(d,k) = 4d$.

Lemma 3. *Given a $g(d,k)(d+\frac{n}{k})$-time bounded base algorithm for a graph with unknown number of nodes n, given depth $d = ab+c$, $a,b,c \in \mathbb{N}$, and k exploring robots, then the Yo* algorithm can explore such a tree within the following number of rounds*

$$\left(d+\frac{n}{k}\right)\left(8g(2a,k)\log k + g(c,k) + 2b(\log k + \log n) + 4\log n\right) .$$

Algorithm 3: The Yo* algorithm using a base algorithm

All k robots start at the root of the tree
Explore the subtree of depth c with the base algorithm
for $j \leftarrow 1, \ldots, b$ **do**
  $R \leftarrow$ set of nodes in depth $\max\{0, c+(j-2)a\}$, which are ancestors to at least one unexplored succeeding node in depth $[c+(j-1)a, c+ja]$
  while $R \neq \emptyset$ **do**
    if $k \leq |R|$ **then**
      Equally partition all nodes in R into sets $V_1, \ldots, V_k$ such that $|V_i| \leq \left\lceil \frac{|R|}{k} \right\rceil$
      for $i \leftarrow 1, \ldots, k$ **do**
        $T_i \leftarrow$ minimum tree connecting all unexplored nodes in depth $[c+(j-1)a, c+ja]$ with an ancestor in V_i
      end for
      while less than $k/2$ subtrees of R are explored **do**
        for all $i \leftarrow 1, \ldots, k$ **do in parallel**
          Perform a DFS exploration step in T_i with robot i
        end for
      end while
    else
      for $i \leftarrow 1, \ldots, |R|$ **do**
        Equally assign k_i robots to node v_i of R such that $k_i \in \left\{ \left\lfloor \frac{k}{|R|} \right\rfloor, \left\lceil \frac{k}{|R|} \right\rceil \right\}$
        $T_i \leftarrow$ minimum tree connecting all unexplored nodes in depth $[c+(j-1)a, c+ja]$ with ancestor v_i
      end for
      while less than $k/2$ subtrees of R are fully explored **do**
        for all $i \leftarrow 1, \ldots, |R|$ **do in parallel**
          Perform one step of the base exploration algorithm on T_i with k_i robots
        end for
      end while
    end if
    $R \leftarrow$ set of nodes in depth $\max\{0, c+(j-2)a\}$, which are ancestors to at least one unexplored succeeding node in depth $[c+(j-1)a, c+ja]$
  end while
end for

Proof. We denote by n_0 the number of nodes in depth at most c. By n_j we denote the number of nodes of the tree with depth in the interval $[1 + c + (j-1)a, c + ja]$. By definition $\sum_{j=0}^{b} n_j = |V|$.

We use the base algorithm to explore the first segment, which needs at most

$$t_1 = g(c,k)\left(d + \frac{n_0}{k}\right) \tag{1}$$

rounds.

In all other rounds we use the base algorithm several times when $k > |R|$. After each iteration of the while-loop from lines 21-25 the number of $|R|$ is reduced by a factor of 2, which implies at most $\log k$ iterations. The variable $\nu = 1, \ldots, \log k$ counts the iterations of this loop. Let $R_{j,\nu}$ be the variable R in the j-th loop and the ν-th iteration. Let $k_{j,\nu}$ be the smallest number of robots in this phase, i.e. $k_{j,\nu} = \lfloor k/|R_{j,\nu}| \rfloor$.

The trees connecting all unexplored nodes with an ancester node in $R_{j,\nu}$ are named $T_{j,\nu,i}$ for $i \in \{1, \ldots, |R_{j,\nu}|\}$. Now define

$$n_{j,\nu} := \text{median}(|V(T_{j,\nu,i})|\ ,\ i \in \{1, \ldots, |R_{j,\nu}|\})$$

where for even number m the median refers to the $m/2$-th largest element. Note that the median implies that

$$n_{j,\nu}\frac{|R_{j,\nu}|}{2} \leq \sum_{i=1}^{|R_{j,\nu}|} |V(T_{j,\nu,i})| \leq n_{j-1} + n_j$$

So, we can conclude that

$$\sum_{\nu=1}^{\log k} n_{j,\nu}|R_{j,\nu}| \leq 2(n_j + n_{j-1}) \log k\ .$$

The run-time of one invocation the base algorithm is by definition at most

$$g(2a, k_{j,\nu})\left(2a + \left\lceil \frac{n_{j,\nu}}{k_{j,\nu}} \right\rceil\right)$$

by design of the loop in line 21. Since $k_{j,\nu} \geq |R_{j,\nu}|$ and $n_{j,\nu} \geq 1$ we can use $\left\lceil \frac{x}{\lfloor y \rfloor} \right\rceil \leq 2\frac{x}{y} + 1$ for $x, y \geq 1$.

$$\left\lceil \frac{n_{j,\nu}}{k_{j,\nu}} \right\rceil = \left\lceil \frac{n_{j,\nu}}{\lfloor k/|R_{j,\nu}| \rfloor} \right\rceil \leq 2\frac{n_{j,\nu}|R_{j,\nu}|}{k} + 1$$

The run-time over all invocations of all these loops is therefore

$$\begin{aligned}
t_2 &\le \sum_{j=1}^{b} \sum_{\nu=1}^{\log k} g(2a, k_{j,\nu})(2a + \lceil n_{j,\nu}/k_{j,\nu} \rceil) \\
&\le g(2a,k) \left(2ab \log k + b \log k + 2 \sum_{j=1}^{b} \sum_{\nu=1}^{\log k} \frac{n_{j,\nu} |R_{j,\nu}|}{k} \right) \\
&\le g(2a,k) \left(2ab \log k + b \log k + 2 \sum_{j=1}^{b} \frac{2(n_{j-1} + n_j) \log k}{k} \right) \\
&\le g(2a,k) \left(2(d-c) \log k + b \log k + 8 \frac{n}{k} \log k \right) \\
&\le \left(2d + b + 8 \frac{n}{k} \right) g(2a,k) \log k \, .
\end{aligned}$$

It remains to count all rebalancing moves of the robots. It takes at most $2d$ steps to reassign a robot to its new tree. For the case $k > |R|$, this iterates at most $b \log k$ times, resulting in

$$t_3 \le 2bd \log k$$

steps.

Now we analyze the case $k \le |R|$. After each iteration of the loop of line 11 the number of nodes in $|R|$ is halved. Hence, the number of loops is bounded by $\log n$. The sum of all iterations of the loop 11 is bounded by $4(n_{j-1} + n_j)/k$ rounds, since $k/2$ robots successfully explore the graph in parallel. So, summing over all j we get

$$t_4 \le \sum_{j=1}^{b} \frac{4(n_{j-1} + n_j)}{k} \le 8 \frac{n}{k} \log n$$

for the DFS-exploration. Again we have to rearrange the robots between the trees which costs at most

$$t_5 \le 2bd \log n$$

additional steps.

So, for $c \le 2a$ and $d \ge 1$ the cost is bounded by:

$$\begin{aligned}
\text{run-time} &= t_1 + t_2 + t_3 + t_4 + t_5 \\
&\le g(c,k) \left(d + \frac{n_0}{k} \right) + g(2a,k) \left(2d + b + 8 \frac{n}{k} \right) \log k \\
&\quad + 2bd \log k + 4 \frac{n}{k} \log n + 2bd \log n \\
&\le d \left(g(c,k) + \left(2 + \frac{b}{d} \right) g(2a,k) \log k \right) \\
&\quad + \frac{n}{k} \left(g(c,k) + 8 g(2a,k) \log k + 4 \log n \right) + 2db(\log k + \log n) \\
&\le \left(d + \frac{n}{k} \right) \left(g(c,k) + 8 g(2a,k) \log k + 2b(\log k + \log n) + 4 \log n \right)
\end{aligned}$$

□

We use polynomials of d for a and b, which results in the following Lemma.

Lemma 4. *Given a $g(d,k)(d+\frac{n}{k})$-time bounded base algorithm for a graph with unknown number of nodes n, given depth d and k exploring robots, then the Yo* algorithm provides a $(d+\frac{n}{k})(9g(2d^\alpha,k)\log k + 8d^{1-\alpha}(\log k+\log n))$-time bounded robot exploration algorithm.*

Proof. We choose $a = \lfloor d^\alpha \rfloor$, $b = \lfloor d/a \rfloor$, and $c = n - ab$. Note that $c \le a$ and $b \le 2d^{1-\alpha}$. From Lemma 3 it follows that the run-time of Yo* is the following:

$$\begin{aligned}
\text{time} &\le \left(d+\frac{n}{k}\right)\left(g(c,k) + 8g(2a,k)\log k + 2b(\log k+\log n) + 4\log n\right)\\
&\le \left(d+\frac{n}{k}\right)\left(g(d^\alpha,k) + 8g(2d^\alpha,k)\log k + 4d^{1-\alpha}(\log k+\log n) + 4\log n\right)\\
&\le \left(d+\frac{n}{k}\right)\left(9g(2d^\alpha,k)\log k + 8d^{1-\alpha}(\log k+\log n)\right)
\end{aligned}$$

Starting from the $4d$-competitive Yo-yo algorithm we choose $\alpha = \frac{1}{2}$ and obtain by the last Lemma a $\mathcal{O}(d^{\frac{1}{2}}(\log k+\log n))$-competitive multi-robot exploration algorithm. This algorithm can be also asymptotically improved by the same lemma. For this we can choose $\alpha = \frac{2}{3}$ and get a $\mathcal{O}(d^{\frac{1}{3}}(\log k)(\log k+\log n))$ algorithm. Of course this process can be iterated using the following lemma.

Lemma 5. *For $k \ge 2$, $c \ge 4$, $\beta \in [0,1]$, $\gamma \ge 0$ and a base exploration algorithm with a run-time of $\left(d+\frac{n}{k}\right) cd^\beta(\log k)^\gamma(\log k+\log n)$, the Yo-star algorithms can achieve an exploration time bound of $\left(d+\frac{n}{k}\right) 20c\cdot d^{\beta/(\beta+1)}(\log k)^{\gamma+1}(\log k+\log n)$.*

Proof. We choose $\alpha = \frac{1}{1+\beta}$ such that $\alpha\beta = 1-\alpha$. This observation will be used for the run-time of an iteration the Yo* algorithm.

$$\begin{aligned}
\frac{\text{time}}{d+\frac{n}{k}} &\le 9g(2d^\alpha,k)\log k + 8d^{1-\alpha}(\log k+\log n)\\
&\le 9c2^\beta d^{\alpha\beta}(\log k)^{\gamma+1}(\log k+\log n) + 8d^{1-\alpha}(\log k+\log n)\\
&\le 18cd^{\beta/(\beta+1)}(\log k)^{\gamma+1}(\log k+\log n) + 8d^{\beta/(\beta+1)}(\log k+\log n)\\
&\le 20cd^{\beta/(\beta+1)}(\log k)^{\gamma+1}(\log k+\log n)\ ,
\end{aligned}$$

where we use $2^\beta \le 2$ and $18c+8 \le 20c$ for $c \ge 4$. □

Note that the iteration $\beta \mapsto \beta/(\beta+1)$ with starting point $\beta = 1$ results in the series $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$. Let $\beta_1 := 1$ and $\beta_{i+1} := \beta_i/(\beta_i+1)$. If $\beta_i = \frac{1}{i}$, then

$$\beta_{i+1} = \frac{\frac{1}{i}}{\frac{1}{i}+1} = \frac{1}{i+1}\ .$$

So, after ℓ iterations of the Yo-star algorithm, starting from the Yo-yo algorithm we have the following ratio:

$$\frac{\text{time}}{\frac{n}{k}+d} \ \le\ 4\cdot 20^\ell d^{\frac{1}{\ell+1}}(\log k)^\ell(\log k+\log n)\ .$$

So far, we have assumed to know the depth. This is not necessary, since we use exponential doubling to find it. This introduces an additional factor of $\log d$ which we will take into account from now on. This competitive factor is only taken into account once, since after a correct guess the recursive approach invokes the next exploration algorithm with the correct depth value.

Theorem 1. *The Yo-star multi-robot exploration algorithm with ℓ iterations can explore an unknown tree with depth d and size n with a competitive ratio of at most*

$$\mathcal{O}\left(20^\ell d^{\frac{1}{\ell+1}}(\log k)^\ell(\log k+\log n)(\log d)\right) .$$

Proof. Since the depth of tree is unknown we iteratively restart the Yo-star exploration algorithm with an assumed depth of $d' = 1, 2, 4, \ldots$. An exploration is canceled if a node with depth larger than d' has been found, then the exploration starts from scratch. In the final step the time for the exploration is therefore at most (assuming $d' = 2d - 1$ in the worst case)

$$\left(2d-1+\frac{n}{k}\right) 4\cdot 20^\ell(2d-1)^{\frac{1}{\ell+1}}(\log k)^\ell(\log k+\log n) .$$

Now $2d - 1 + n/k \leq 2(d + n/k)$ and $(2d-1)^{\frac{1}{\ell+1}} \leq (2d)^{\frac{1}{\ell+1}} \leq 2d^{\frac{1}{\ell+1}}$ results in an additional factor of 4. It takes $\log d$ iterations until $d' \geq d$ and therefore we have a total run-time of at most

$$16\left(d+\frac{n}{k}\right)\cdot 20^\ell d^{\frac{1}{\ell+1}}(\log k)^\ell(\log k+\log n)\log d$$

The competitive factor originates from the observation that the minimal time for offline exploration is $\max\{d, n/k\} \geq \frac{1}{2}(d + n/k)$. □

This is the main result of this paper. What follows is a discussion of how many iterations are necessary to achieve best possible asymptotical bounds. It turns out that the relationship between the depth and the number of robots is crucial. If $d = \mathcal{O}((\log k)^c)$ then already the Yo-yo algorithm provides a competitive ratio of $\mathcal{O}((\log k)^c)$. If the depth is larger with respect to the number of robots, then Yo* provides better bounds.

Theorem 2. *The Yo-star algorithm can achieve a competitive factor of*

$$2^{(2+o(1))\sqrt{(\log d)(\log\log k)}}(\log k)(\log k+\log n)$$

for a k-multi-robot exploration of graphs of size n and depth d.

Proof. Again we test the depth of the tree by performing the ℓ iterations of the Yo* algorithm, where we double a depth parameter d' every time we finde a node in depth $d' + 1$. Then, we relaunch the exploration. As the iteration depth of Yo* we choose $\ell = \left\lceil\sqrt{\frac{\log d'}{\log\log k}}\right\rceil$, because

$$(\log k)^\ell = 2^{\left\lceil\sqrt{\frac{\log d'}{\log\log k}}\right\rceil\log\log k} \leq 2^{\sqrt{(\log d')(\log\log k)}}\log k$$

and

$$d'^{2/(2\ell+1)} \leq 2^{(\log d)\sqrt{\log\log k}/\sqrt{\log d}} = 2^{\sqrt{(\log d)(\log\log k)}}$$

Now

$$20^{\ell}(\log d') \leq 2^{\sqrt{\log d}+\log\log d} = 2^{o(1)}\sqrt{(\log d)(\log\log k)}$$

for large enough k. So, the only remaining relevant factor is $\log k + \log n$ which implies the result. □

This bound is not always smaller than the best known competitive ratio of $\mathcal{O}(k/\log k)$. Yet, for trees with depth $d = 2^{(\frac{1}{4}-\epsilon)\frac{(\log k)^2}{\log\log k}}$, $\epsilon > 0$ and size $n = 2^{\mathcal{O}(2^{\sqrt{d}})}$, which includes the interesting cases of $d = \mathcal{O}(k^c)$ and $n = 2^{d^{o(1)}}$ for any $c > 0$, the Yo-star algorithms is currently the best available multi-robot exploration algorithm. See the appendix for a proof of this observation.

4 Conclusions

We discussed in this work the collaborative multi-robot exploration of trees. The algorithms know the positions of all robots and are not restricted in computation. Until now a wide gap was open between upper bound of $\mathcal{O}(\frac{k}{\log k})$ and the lower bound of $\Omega(\frac{\log k}{\log\log k})$. While these bounds could not been improved for nearly a decade and the only improvements have been happening on more restricted models, we finally were able to present new upper borders for the collective tree exploration.

The first, rather simple, Yo-yo algorithm has a competitive ratios of $4d$ and improves this bound for $d = o(k/\log k)$ robots. If the number of robots is smaller, then our Yo-star algorithm provides a new bound of

$$2^{(2+o(1))\sqrt{(\log d)(\log\log k)}}(\log k)(\log k + \log n)\ .$$

This hard to understand bound needs some interpretation and one can be derived the bounds for $k \leq n$ shown in Table 1 for any constant $c > 0$. A proof of the results in this table is given in the Appendix.

k	d	n	competitive factor	algorithm
$2^{d^{1/c}}$	$(\log k)^c$		$4(\log k)^c$	Yo-yo
$d^{\Omega(1)}$	$k^{\mathcal{O}(1)}$		$k^{o(1)}\log n$	Yo*
$d^{\Omega(1)}$	$k^{\mathcal{O}(1)}$	$2^{d^{o(1)}}$	$k^{o(1)}$	Yo*
	$2^{\frac{c^2}{4}(\log k)^2/\log\log k}$		$k^{c(1+o(1))}\log n$	Yo*
$2^{\omega(\sqrt{\log d\log\log d})}$		$2^{O(2^{\sqrt{\log d}})}$	$k^{o(1)}$	Yo*
		$2^{2^{\mathcal{O}(\sqrt{(\log d)(\log\log k)})}}$	$2^{\mathcal{O}(\sqrt{(\log d)\log\log k})}$	Yo*

Table 1. Competitive exploration time ratios for the Yo-yo and the Yo* algorithm

We presented in this work the first collaborative exploration for trees reaching subpolynomial competitive ratio of $o(k^{\epsilon})$ for any constant $\epsilon > 0$ and $k < n$, especially

this is the case for the Jellyfish tree with $k = d$ and $n/k = d$, which is used to establish the lower bound of $\mathcal{O}(\log k/ \log \log k)$ [10]. This is an important step closing the gap between the upper and lower bound. Future work will be dedicated to the multi-robot exploration of general graphs using the Yo-yo and Yo* approach.

5 Acknowledgments

We are very grateful for the comments of the anonymous reviewers pointing out numerous mistakes in the first version.

References

1. S. Albers and M. R. Henzinger. Exploring Unknown Environments. *SIAM Journal on Computing*, 29(4):1164, 2000.
2. S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, pages 842–843, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.
3. M. A. Bender. The power of team exploration: Two robots can learn unlabeled directed graphs. In *In Proceedings of the Thirty Fifth Annual Symposium on Foundations of Computer Science*, pages 75–85, 1994.
4. M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, 176(1):1 – 21, 2002.
5. P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *Robotics, IEEE Transactions on*, 27(4):707–717, Aug 2011.
6. X. Deng and C. Papadimitriou. Exploring an unknown graph. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 355 –361 vol. 1, oct 1990.
7. D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznanski. Fast collaborative graph exploration. In *Automata, Languages, and Programming*, volume 7966 of *Lecture Notes in Computer Science*, pages 520–532. Springer Berlin Heidelberg, 2013.
8. A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theor. Comput. Sci.*, 326:343–362, October 2004.
9. M. Dynia, J. Kutylowski, F. Heide, and C. Schindelhauer. Smart robot teams exploring sparse trees. In *Mathematical Foundations of Computer Science 2006*, volume 4162 of *Lecture Notes in Computer Science*, pages 327–338. Springer Berlin Heidelberg, 2006.
10. M. Dynia, J. Lopuszanski, and C. Schindelhauer. Why robots need maps. In *Proceedings of the 14th international conference on Structural information and communication complexity*, SIROCCO'07, pages 41–50, Berlin, Heidelberg, 2007. Springer-Verlag.
11. R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In G. Brodal and S. Leonardi, editors, *Algorithms – ESA 2005*, volume 3669 of *Lecture Notes in Computer Science*, pages 11–22. Springer Berlin / Heidelberg, 2005. 10.1007/11561071_4.
12. P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Netw.*, 48:166–177, October 2006.
13. K.-T. Förster and R. Wattenhofer. Directed graph exploration. In R. Baldoni, P. Flocchini, and R. Binoy, editors, *Principles of Distributed Systems*, volume 7702 of *Lecture Notes in Computer Science*, pages 151–165. Springer Berlin Heidelberg, 2012.
14. Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24(3):197–224, Apr. 2003.

15. Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization*, pages 1–16, 2012.
16. A. Kolenderska, A. Kosowski, M. Malafiejski, and P. Zylinski. An improved strategy for exploring a grid polygon. In S. Kutten and J. Őerovnik, editors, *Structural Information and Communication Complexity*, volume 5869 of *Lecture Notes in Computer Science*, pages 222–236. Springer Berlin Heidelberg, 2010.
17. C. Ortolf and C. Schindelhauer. Online multi-robot exploration of grid graphs with rectangular obstacles. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 27–36, New York, NY, USA, 2012. ACM.
18. N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410:1–58, Oak Ridge National Laboratory, July 1993.