Brief Announcement: Efficient Collaborative Tree Exploration with Breadth-First Depth-Next

Romain Cosson
Inria
Paris, France
romain.cosson@inria.fr

Laurent Massoulié
Inria
Paris, France
laurent.massoulie@inria.fr

Laurent Viennot
Inria
Paris, France
laurent.viennot@inria.fr

ABSTRACT

We consider the problem of *collaborative tree exploration* posed by Fraigniaud, Gasieniec, Kowalski, and Pelc [8] where a team of k agents is tasked to collectively go through all the edges of an unknown tree as fast as possible and return to the root. Denoting by n the total number of nodes and by D the tree depth, the $O(n/\log(k) + D)$ algorithm of [8] achieves the best competitive ratio known with respect to the optimal exploration algorithm that knows the tree in advance, which takes order $\max\{2n/k, 2D\}$ rounds. Brass, Cabrera-Mora, Gasparri, and Xiao [1] consider an alternative performance criterion, the additive overhead with respect to $2n/k$, and obtain a $2n/k + O((D+k)^k)$ runtime guarantee. In this announcement, we present 'Breadth-First Depth-Next' (BFDN), a novel and simple algorithm that performs collaborative tree exploration in time $2n/k + O(D^2 \log(k))$, thus outperforming [1] for all values of (n, D) and being order-optimal for fixed k and trees with depth $D = o(\sqrt{n})$. The proof of our result crucially relies on the analysis of a simple two-player game with balls in urns that could be of independent interest. We extend the guarantees of BFDN to: scenarios with limited memory and communication, adversarial setups where robots can be blocked, and exploration of classes of non-tree graphs. Finally, we provide a recursive version of BFDN with a runtime of $O_\ell(n/k^{1/\ell} + \log(k) D^{1+1/\ell})$ for parameter $\ell \geq 1$, thereby improving performance for trees with large depth. A complete version of the paper is available online [2].

CCS CONCEPTS

• **Theory of computation → Online algorithms**; **Distributed algorithms**; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

collaborative exploration, graphs, trees, depth, adversarial game

ACM Reference Format:
Romain Cosson, Laurent Massoulié, and Laurent Viennot. 2023. Brief Announcement: Efficient Collaborative Tree Exploration with Breadth-First Depth-Next. In *ACM Symposium on Principles of Distributed Computing (PODC '23), June 19–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583668.3594568

PODC '23, June 19–23, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0121-4/23/06…$15.00
https://doi.org/10.1145/3583668.3594568

1 INTRODUCTION

Problem Setting. A team of k agents, or robots, is tasked to collectively traverse all the edges of a tree as fast as possible and then return to the root. At initialization, all robots are located at the root. At each round, the robots move synchronously along one incident edge to reach a neighbour. Edges are revealed along with their unique identifier when a robot reaches one of their endpoints. Following [8], we consider two communication models. The *complete communication* model, in which communication is unrestricted, and consequently the robots share a map of the explored sub-tree. The *write-read communication* model, in which robots communicate through whiteboards that are located at all nodes.

Related Works. In the case of a single robot, the "Depth First Search" (DFS) algorithm is optimal for traversing the edges of a tree. It can be implemented both *offline* (the tree is known in advance) and *online* (edges are revealed when reached). One way to describe DFS in an online fashion is to have the robot go through an adjacent unexplored edge if possible and go up towards the root otherwise. After $2(n-1)$ rounds, where n is the number of nodes, all edges have been traversed twice and the robot is at the root.

In the multi-robot setting, with $k \geq 2$, traversing all the edges of a tree in an *offline* manner requires order $\max\{2n/k, 2D\} \sim n/k + D$ synchronous rounds [5, 11]. Unfortunately, no *online* exploration algorithm can attain this performance [7]. The cost of an online exploration algorithm is usually quantified through its *competitive ratio*. An algorithm $\mathcal{A}_k$ using $k \geq 2$ robots, has competitive ratio $c(k)$ if it can explore any tree with n nodes and depth D in $c(k)(n/k + D)$ rounds. The algorithm CTE [8] has a runtime of $O(n/\log(k) + D)$ and the best known competitive ratio of $O(k/\log(k))$.

The limited progress of competitive analysis for collaborative tree exploration led most subsequent works to study algorithms with super-linear runtime in n or D [1, 3, 4, 6, 9, 11]. In this spirit, [11] proposed a recursive algorithm called Yo* that runs with complete communication in $O(2^{O(\sqrt{\log D \log\log k})} \log(k)(\log(k)+\log(n))(n/k + D))$ rounds. On the other hand, [1] proposed a novel analysis of CTE yielding a guarantee of $\frac{2n}{k} + O((k+D)^k)$, thus with optimal dependence in n and a large additive cost which does not depend on n. The algorithm we propose with its guarantee of $\frac{2n}{k} + O(D^2 \log(k))$ falls in this line of work. Our guarantee improves over [1] for all values of (n, D, k), and improves upon CTE and Yo* in specific ranges of parameters as depicted in Figure 1.

Main Results. In this paper, we present a simple and novel algorithm that achieves collaborative tree exploration with k agents in $\frac{2n}{k} + D^2(\min\{\log(k), \log(\Delta)\} + 2)$ rounds for any tree with n nodes, depth D and maximum degree Δ. Our algorithm can be implemented in the write-read communication model.

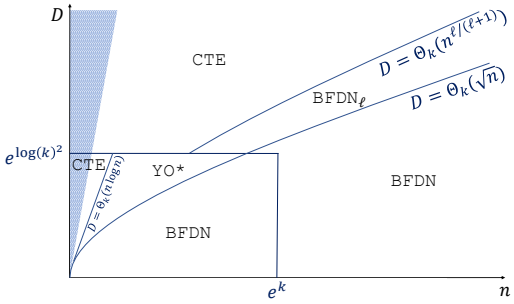

Figure 1: Regions of parameters $n \leq D$ where either of CTE, Yo*, BFDN and BFDN$_\ell$ has the best runtime guarantee.

The algorithm is called "Breadth-First Depth-Next" (BFDN). It is first presented in the complete communication setting in Section 2. The behaviour of the robots can be summarized as follows: when located at the root, robots are sent to a node which is adjacent to the highest unexplored edge (as in a breadth-first search). Upon arrival, robots change behaviour until they reach the root again: they will go through an unexplored edge if some is adjacent and will go up towards the root otherwise (as in a depth-first search).

The analysis of this algorithm involves a simple zero-sum two-player game where a player and an adversary move k balls in k urns. We describe in Section 3 a strategy that induces a cost of at most $k \min\{\log(k), \log(\Delta)\} + k$ to the player.

Our algorithm can benefit from several extensions briefly described in Section 4, i) exploration of specific classes of non-tree graphs, ii) scenarios with constrained communications and memory, iii) setups where an adversary chooses at each time step which robots are allowed to move iv) improved dependence in D, with a recursive version of BFDN in $O_\ell\left(\frac{n}{k^{1/\ell}} + \min\{\log(k), \log(\Delta)\} D^{1+1/\ell}\right)$ where $\ell \geq 1$ is some constant provided as input.

Notations. The notation $\log(\cdot)$ refers to the natural logarithm. For some integer $k \in \mathbb{N}$ we will use the abbreviation $[k] = \{1, \dots, k\}$.

A tree $T = (V, E)$ is rooted at some specific node denoted $\texttt{root} \in V$ from which all robots start the exploration. For a node $v \in V$, $\delta(v)$ is the distance of v to the root and $T(v)$ denotes the sub-tree of T rooted at v containing all the descendants of v. The depth of T is $D = \max_{v \in V} \delta(v)$. We will also use a notion of *partially explored tree* (defined in Section 2) that enjoys the same definitions.

2 BREADTH-FIRST DEPTH-NEXT

Our main result on BFDN, described below, is the following.

Theorem 2.1. BFDN *achieves online exploration of any tree with k robots in at most*

$$\frac{2n}{k} + D^2(\min\{\log(\Delta), \log(k)\} + 2)$$

rounds, where Δ is the maximum degree of the tree, n is the number of nodes, and D is the depth.

Following [8], we first consider a synchronous model with all-to-all communications. The team of k robots operates in rounds, and the runtime of an exploration algorithm is measured by the number of rounds before termination.

Partially Explored Tree. At a given round of exploration, V denotes the set of *explored nodes*, i.e. nodes that have been occupied by at least one robot in the past, and E denotes the set of *discovered edges*, i.e. edges that have at least one explored endpoint. The discovered edges which have exactly one explored endpoint are called *dangling edges*. Such edges can be viewed as a pair $(u, ?)$, with $u \in V$. The *partially explored tree* $T_{\text{online}} = (V, E)$ thus encodes all the information gathered by the robots at some points of the exploration. If there are no dangling edges in T_{online}, it means that exploration is complete and that the partially explored tree equals the underlying tree $T_{\text{offline}} \in \mathcal{T}(n, D)$.

Collaborative Exploration Algorithm. A collaborative exploration algorithm under the complete communication model is formally defined as a function that maps a partially explored tree $T = (V, E)$, the current positions of the agents $p_1, \dots, p_k \in V^k$ and all past movements of the agents to a list of *selected edges* $e_1, \dots, e_k \in (E \cup \{\perp\})^k$ that will be traversed in the current round. The selected edges $e_i \in E$ must be adjacent to the positions p_i. By convention, $\perp$ indicates that the corresponding agent will not move at the next round. In pseudo-code, the routine $\texttt{SELECT}(\texttt{Robot}_i, e)$ performs the assignment $e_i \leftarrow e$. When all agents have selected a next move, the routine MOVE is applied and all agents move along their selected edge synchronously. The partially explored tree (V, E) is updated with the new information provided by the agents that have traversed a dangling edge. For any rooted tree, exploration starts with $V = \{\texttt{root}\}$ and E the set of edges dangling at the root. The collaborative exploration algorithm is iterated until (V, E) contains no dangling edges and the position of all agents is back at the root.

Breadth-First Depth-Next Algorithm. We provide a brief description of BFDN, Algorithm 1. When located at the root, $\texttt{Robot}_i$ with $i \in [k]$ is assigned an *anchor* $v_i \in V$ which is an explored node that is adjacent to at least one dangling edge. If no such node exists, the anchor is the root itself. The exact anchor assignment is specified by procedure Reanchor which gives the priority to nodes that are the closest to the root and that have the least number of anchored robots. $\texttt{Robot}_i$ then attains this anchor in a series of breadth-first moves performed with procedure BF. When the anchor is reached, the robot only makes depth-next moves until it returns to the root with procedure DN. In a sequence of depth-next moves, the robot always goes through a dangling edge if one is available (i.e. adjacent and not already selected as next move by another robot), and goes one step up towards the root otherwise. This will result in a depth-first-like exploration inside $T(v_i)$ followed by a direct travel back from v_i to the root. The algorithm stops when all robots are at the root and cannot be reassigned a new anchor because there are no more dangling edges. We conclude this paragraph by stating the following claim that provides intuition on the algorithm.

Claim 1. *At all rounds, all dangling edges, are in $\cup_{i \in [k]} T(v_i)$.*

Proof of Claim 1. Consider some dangling edge e and its explored endpoint $v \in V$. At the round when v was explored by a robot, that robot must have been performing a depth-next move because the depth of its anchor was less than or equal to the depth of v which was adjacent to a dangling edge. Thus, the robot cannot have left $T(v)$ before the edge e was visited. Consequently, that robot is still rooted at some ancestor v_i of v, thus $e \in \cup_{i \in [k]} T(v_i)$. □

Algorithm 1 BFDN "Breadth-First Depth-Next"

Ensure: The robots visit all nodes and return to the root.

V = list of explored nodes ; E = list of discovered edges
$v_i \leftarrow$ root $\forall i \in \{1, \dots, k\}$ ▷ Initialize anchors.
$S_i \leftarrow [\,]\ \forall i \in \{1, \dots, k\}$ ▷ Initialize empty stacks.
do ▷ Round t.
  for $i = 1$ to k **do** ▷ Sequential decisions.
    if Robot$_i$ is at root **then**
      $v_i \leftarrow$ Reanchor(i)
      Stack in S_i the list of edges that lead to v_i
    if S_i is not empty **then**
      BF(i)
    else
      DN(i)
  MOVE robots on their selected edge ▷ Synchronous moves.
while not all robots are at the root

procedure BF(i)
  Unstack $e \in E$ from S_i and SELECT(Robot$_i$, e)

procedure DN(i)
  if Robot$_i$ adjacent to dangling unselected edge $e \in E$ **then**
    SELECT(Robot$_i$, e)
  else
    SELECT(Robot$_i$, up) ▷ If at the root, up is $\perp$.

procedure Reanchor(i)
  $A = \{v \in V$ adjacent to dangling edge with $\delta(v)$ minimal$\}$
  if $A \neq \emptyset$ **then** ▷ Choose anchor of minimum load.
    $v_i \leftarrow \arg\min_{v \in A} \#\{j \ s.t.\ v_j = v\}$
  else ▷ The tree is explored.
    $v_i \leftarrow$ root

3 BALLS IN URNS GAME

In this section we describe a two-player zero-sum board game that is crucial to the analysis of BFDN. An optimal strategy of the game is in Theorem 3.1 and its implication is stated in Corollary 3.2.

Game Description. At time $t \in \mathbb{N}$, the board of the game is a list of k integers $(n_1^t, \dots, n_k^t)$ that represent the load of k urns with a total of k balls. When the game starts at $t = 0$, we have $n_i^0 = 1$ and at every instant t we have $\sum_{i \in [k]} n_i^t = k$ and $n_i^t \geq 0$. At time t, player A (the adversary) chooses an urn $a_t \in [k]$ that is not empty, i.e. such that $n_{a_t}^t \geq 1$, and then player B (the player) chooses an urn $b_t \in [k]$ and moves a ball from urn a_t to urn b_t. At the beginning of time $t+1$, the board thus satisfies $n_{a_t}^{t+1} = n_{a_t}^t - 1$ and $n_{b_t}^{t+1} = n_{b_t}^t + 1$.

Goal of the Game. At a given time t, we denote by U_t the set of urns that have never been selected by the adversary, $U_t = \{1, \dots, k\} \setminus \{a_0, \dots, a_{t-1}\}$. The game stops when all urns in U_t contain at least Δ balls, i.e. $n_i^t \geq \Delta, \forall i \in U_t$. If $\Delta \geq k$, the game stops when all urns have been chosen, i.e. $U_t = \emptyset$. The goal of player B is to end the game as soon as possible, while the goal of the adversary is to play for as long as it can.

Strategy of the Player. At time t, the player picks the urn b_t that contains the least number of balls among the urns that were never chosen by the adversary, i.e. $b_t \in \arg\min_{i \in [k] \setminus \{a_0, \dots, a_t\}} n_i^t$.

Theorem 3.1. *If the player uses the strategy above, the game ends in at most* $k \min\{\log(\Delta), \log(k)\} + k$ *steps.*

Corollary 3.2. *In an execution of* BFDN*, then number of calls to procedure* Reanchor *is at most* $D(k \min\{\log(k), \log(\Delta)\} + k)$.

4 EXTENSIONS OF BFDN

4.1 Restricted Memory and Communications

In [2], we introduce a setting with restricted memory and communication where robots each have $\Delta + D\log(\Delta)$ bits of internal memory and are allowed to communicate with a central planner only when they are located at the root. This setting encompasses the read-write communication model.

Proposition 4.1. *In this communication model,* BFDN *achieves tree exploration in at most* $\frac{2n}{k} + D^2(\min\{\log(k), \log(\Delta)\} + 2)$ *rounds.*

4.2 Adversarial Robot Break-Downs

In [2], we introduce a setting to account for break-downs and varied speeds. At each round $t \in \mathbb{N}$, robot i is allowed to move if some variable $M_{ti} = 1$ and is blocked if $M_{ti} = 0$. We assume $\mathbb{M} = (M_{ti})_{t \in \mathbb{N}, i \in [k]}$ takes arbitrary 0/1 values and denote the average distance travelled $A(\mathbb{M}) = \frac{1}{k} \sum_{t \in \mathbb{N}} \sum_{i \in [k]} M_{ti} \in \mathbb{N} \cup \{+\infty\}$.

Proposition 4.2. *For any sequence* $\mathbb{M} \in \{0,1\}^{\mathbb{N} \times [k]}$ *satisfying* $A(\mathbb{M}) \geq \frac{2n}{k} + D^2(\log(k) + 2)$ *all edges will be visited by* BFDN.

4.3 Exploration of non-tree Graphs

BFDN can be executed on a graph, with a minor modification: a robot rediscovering a node backtracks and "closes" the corresponding edge. In [2], we assume robots always knows their edge-distance to the origin and extend our guarantees to graphs. This assumption holds for grid graphs with rectangular obstacles [10].

Proposition 4.3. *Given such a graph* $G = (V, E)$ *with* m *edges, diameter* D *and maximum degree* Δ*, under the assumption above,* BFDN *runs in* $\frac{2m}{k} + D^2(\min\{\log(\Delta), \log(k)\} + 2)$ *rounds.*

4.4 Recursive Variant

In [2] we present a recursive variant of BFDN, denoted BFDN$_\ell$ achieving a better dependence in D though worse dependence in n. The variant works by dividing the robots in teams running independent instances of BFDN. The technique is applied recursively, with $\ell \in \mathbb{N}$ denoting the depth of the recursive stack.

Proposition 4.4. *For any integer* $\ell \geq 1$, BFDN$_\ell$ *explores a tree with* n *nodes, depth* D*, maximum degree* Δ *with* k *robots in* $\frac{4n}{k^{1/\ell}} + 2^{\ell+1}(\ell + 1 + \min\{\log(\Delta), \log(k)/\ell\})\, D^{1+1/\ell}$ *rounds.*

ACKNOWLEDGMENTS

The authors thank the reviewers for their careful remarks and the entire Argo team for enlightening discussions. RC thanks Pierre Fraigniaud for valuable feedback and Maxime Cartan for Python demo (at https://github.com/Romcos/BFDN). Work supported by PRAIRIE ANR-19-P3IA-0001 and Tempogral ANR-22-CE48-0001.

REFERENCES

[1] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. 2011. Multirobot Tree and Graph Exploration. *IEEE Trans. Robotics* 27, 4 (2011), 707–717. https://doi.org/10.1109/TRO.2011.2121170

[2] Romain Cosson, Laurent Massoulié, and Laurent Viennot. 2023. Breadth-First Depth-Next: Optimal Collaborative Exploration of Trees with Low Diameter. *arXiv preprint arXiv:2301.13307* (2023).

[3] Dariusz Dereniowski, Yann Disser, Adrian Kosowski, Dominik Pajak, and Przemyslaw Uznanski. 2013. Fast Collaborative Graph Exploration. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 7966)*, Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg (Eds.). Springer, 520–532. https://doi.org/10.1007/978-3-642-39212-2_46

[4] Yann Disser, Frank Mousset, Andreas Noever, Nemanja Skoric, and Angelika Steger. 2017. A General Lower Bound for Collaborative Tree Exploration. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10641)*, Shantanu Das and Sébastien Tixeuil (Eds.). Springer, 125–139. https://doi.org/10.1007/978-3-319-72050-0_8

[5] Miroslaw Dynia, Miroslaw Korzeniowski, and Christian Schindelhauer. 2006. Power-Aware Collective Tree Exploration. In *Architecture of Computing Systems - ARCS 2006, 19th International Conference, Frankfurt/Main, Germany, March 13-16, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3894)*, Werner Grass, Bernhard Sick, and Klaus Waldschmidt (Eds.). Springer, 341–351. https://doi.org/10.1007/11682127_24

[6] Miroslaw Dynia, Jaroslaw Kutylowski, Friedhelm Meyer auf der Heide, and Christian Schindelhauer. 2006. Smart Robot Teams Exploring Sparse Trees. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4162)*, Rastislav Kralovic and Pawel Urzyczyn (Eds.). Springer, 327–338. https://doi.org/10.1007/11821069_29

[7] Miroslaw Dynia, Jakub Lopuszanski, and Christian Schindelhauer. 2007. Why Robots Need Maps. In *Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4474)*, Giuseppe Prencipe and Shmuel Zaks (Eds.). Springer, 41–50. https://doi.org/10.1007/978-3-540-72951-8_5

[8] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. 2006. Collective tree exploration. *Networks* 48, 3 (2006), 166–177. https://doi.org/10.1002/net.20127

[9] Yuya Higashikawa, Naoki Katoh, Stefan Langerman, and Shin-ichi Tanigawa. 2014. Online graph exploration algorithms for cycles and trees by multiple searchers. *J. Comb. Optim.* 28, 2 (2014), 480–495. https://doi.org/10.1007/s10878-012-9571-y

[10] Christian Ortolf and Christian Schindelhauer. 2012. Online multi-robot exploration of grid graphs with rectangular obstacles. In *24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '12, Pittsburgh, PA, USA, June 25-27, 2012*, Guy E. Blelloch and Maurice Herlihy (Eds.). ACM, 27–36. https://doi.org/10.1145/2312005.2312010

[11] Christian Ortolf and Christian Schindelhauer. 2014. A Recursive Approach to Multi-robot Exploration of Trees. In *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8576)*, Magnús M. Halldórsson (Ed.). Springer, 343–354. https://doi.org/10.1007/978-3-319-09620-9_26