Collective Tree Exploration via Potential Function Method

Romain Cosson, Laurent Massoulié

Collective Tree Exploration via Potential Function Method

Romain Cosson
Inria, Paris, France

Laurent Massoulié
Inria, Paris, France

Abstract

We study the problem of collective tree exploration (CTE) in which a team of k agents is tasked to traverse all the edges of an unknown tree as fast as possible, assuming complete communication between the agents [14]. In this paper, we present an algorithm performing collective tree exploration in $2n/k + \mathcal{O}(kD)$ rounds, where n is the number of nodes in the tree, and D is the tree depth. This leads to a competitive ratio of $\mathcal{O}(\sqrt{k})$, the first polynomial improvement over the $\mathcal{O}(k)$ ratio of depth-first search. Our analysis holds for an asynchronous generalization of collective tree exploration. It relies on a game with robots at the leaves of a continuously growing tree extending the "tree-mining game" of [6] and resembling the "evolving tree game" of [3]. Another surprising consequence of our results is the existence of algorithms $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$ for layered tree traversal (LTT) with cost at most $2L/k + \mathcal{O}(kD)$, where L is the sum of all edge lengths. For the case of layered trees of width w and unit edge lengths, our guarantee is thus in $\mathcal{O}(\sqrt{w}D)$.

2012 ACM Subject Classification Theory of computation → Online algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases collective exploration, online algorithms, evolving tree, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.35

Funding This work was supported by PRAIRIE ANR-19-P3IA-0001.

Acknowledgements The authors thank Laurent Viennot for stimulating discussions, the Argo team at Inria, as well as the anonymous reviewers for their suggestions.

1 Introduction

The present study concerns collective tree exploration (CTE), a problem introduced in the field of distributed computing by [13]. The goal is for a team of agents or robots, initially located at the root of an unknown unweighted tree, to go through all of its all edges as quickly as possible before returning to the root. At all rounds, each robot moves along one edge to reach a neighboring node. When a robot attains the endpoint of a new edge, the existence of that edge is revealed to the entire team. Following the centralized full-communication setting, we assume that robots can communicate and compute at no cost. They thus share at all times a map of the explored sub-tree and of the unexplored edges at its boundary. In collective tree exploration, the number of nodes of the tree is denoted by n and the tree depth is denoted by D, both quantities are unknown initially.

Another seemingly unrelated problem is layered graph traversal (LGT). It was introduced in the literature on online algorithms by [16]. We describe the problem for the special case of trees, which was extensively studied [12, 17, 3]. The goal is for a single agent, initially located at the origin of an unknown tree with non-negative edge lengths, to reach another node (called the target), while enduring a small movement cost, in the sense of the total travelled length. The set of nodes that are i hops away from the origin is called the i-th layer, the maximum cardinality of a layer is called the width and is denoted by w, the length

of the path from the origin to the target is called the depth and is denoted by D, and the sum of all edge lengths is denoted by L. The tree is revealed iteratively: at step i all nodes and edges up to layer i are revealed, and the agent must move to some node in the i-th layer. This repeats until the layer containing the target is revealed.

Main results. In this paper we present a deterministic algorithm performing collective tree exploration (CTE) with k robots in $2n/k + \mathcal{O}(kD)$ synchronous rounds for any tree with n nodes and depth D. This algorithm, when used by a fraction $k' = \lfloor \sqrt{k} \rfloor$ robots while the remaining $k - k'$ robots stay idle at the root, achieves a competitive ratio of order $\mathcal{O}(\sqrt{k})$. The algorithm guarantee is derived for the more general asynchronous setting (ACTE) defined in Section 3, making it applicable to a wider range of real-world scenarios.

Our analysis relies on a two-player game, that we call the "continuous tree-mining game" (CTM). In this game, the adversary controls the continuous evolution of a tree while the player controls the position of k "miners" located at its leaves. The game differs from the "evolving tree game" of [3] in that the player may block the extension of a leaf of the tree by attributing it a single miner. We show that it is possible for the player of this game to get all miners to reach depth D with a total movement cost of at most $\mathcal{O}(k^2 D)$.

Another consequence of our analysis is a sequence of randomized algorithms $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$ for layered tree traversal (LTT) which satisfy for any $k \in \mathbb{N}$ that the cost of algorithm $\mathcal{A}_k$ on a layered tree T of depth D and length L is bounded by $2L/k + \mathcal{O}(kD)$. This result highlights for the first time a strong connection between collective tree exploration and layered graph traversal, which were until now introduced and studied by two distinct fields. For the case of layered trees of width w and unit edge lengths, since $L \leq wD$, our guarantee is thus in $\mathcal{O}(\sqrt{w}D)$. To the best of our knowledge, this is the first algorithm to improve over the trivial $\mathcal{O}(L)$ guarantee for that problem.

Background on Collective Tree Exploration. The problem of collective tree exploration has a rich history in the field of distributed algorithms and robotics. It was introduced by [13] along with two communication models. A centralized "complete communication" model, in which communications are unrestricted, which we study in this paper ; and a distributed "write-read communication" model in which agents communicate through whiteboards located at all nodes. A collective exploration algorithm is said to be order $c(k)-$competitive if its runtime on a tree with n nodes and of depth D is bounded by $\mathcal{O}\left(c(k)\left(\frac{n}{k} + D\right)\right)$ (see [14]). A competitive ratio of $\mathcal{O}(k)$ is thus trivially achieved by a single depth-first search. [14] proposed a $\mathcal{O}(k/\log(k))$-competitive algorithm which can be implemented in the distributed communication model, and thus also in the complete communication model. They suggested that a constant ratio could be achieved in the complete communication model (see initial version [13]). This conjecture was disproved by [11] who showed that the competitive ratio of any deterministic algorithm is at least in $\Omega(\log(k)/\log\log(k))$. Many works followed, tackling diverse questions such as: quasi-linear algorithms [15], power constraints [10], and the case of many explorers $k \gg n$ [9]. A new type of competitive analysis was proposed by [2], with a guarantee of the form $2n/k + \mathcal{O}((D+k)^k)$, where $2n/k$ is a lower-bound on the time required by the robots to traverse all edges and return to the origin and where we can thus call the quantity $\mathcal{O}((D+k)^k)$ the "competitive overhead" or "penalty". This quantity was improved to $\mathcal{O}(\log(k)D^2)$ by a simple algorithm combining breadth-first search and depth-first search [7]. The approach was then recently used by [6] to slightly improve the competitive ratio of collective tree exploration in the complete communication model to order $\mathcal{O}(k/\exp(\sqrt{\ln 2 \ln k}))$, using a linear in-$D$ competitive overhead of $\mathcal{O}(k^{\log_2(k)-1}D)$.

These analyses were performed in the complete communication model for an asynchronous extension of collective tree exploration (ACTE). The competitive ratio presented in this paper, in $\mathcal{O}(\sqrt{k})$, is the first to display a polynomial improvement over the aforementioned $\mathcal{O}(k)$ ratio of depth-first search. The present discussion is summarized in Table 1.

Table 1 Previous work on competitive analysis of collective tree exploration (CTE), in the complete communication model. All results hold up to a multiplicative constant.

$\mathcal{O}(\cdot)$ Runtime	Competitive Ratio $c(\cdot)$ $c(k)(\frac{n}{k}+D)$	Competitive Overhead $f(\cdot,\cdot)$ $\frac{2n}{k}+f(k,D)$
[13]	$k/\ln k$	-
[2]	-	$(D+k)^k$
[7]	-	$\ln kD^2$
[6]	$k/\exp(\sqrt{\ln 2 \ln k})$	$k^{\log_2 k-1}D$
This work	$\sqrt{k}$	kD

Background on Layered Graph Traversal. The problem has a rich history in the field of online algorithms. It was first described in a paper by Papadimitriou and Yannanakis [16] titled "Shortest path without a map" analyzing the case of layered graph with width $w = 2$. Independently the problem was introduced by [5] under the denomination "Metrical Service System", which was inspired by [1]. The equivalence, between both settings, was noticed by [12]. They also introduced the denomination Layered Graph Traversal (LGT) and observed that its competitive analysis can be reduced to the special case of Layered Tree Traversal (LTT). For arbitrary width w and depth D, they proposed a deterministic algorithm with cost bounded by $\mathcal{O}(9^w D)$, i.e. of competitive ratio $\mathcal{O}(9^w)$. The quantity was later improved to $\mathcal{O}(w2^w)$ by [4], nearly matching the lower-bound in $\Omega(2^w)$ [12]. Using a randomized algorithm, [3] obtained a $\mathcal{O}(w^2)$ competitive ratio, nearly matching the randomized lower bound in $\Omega(w^2/\log(w))$ [17]. The approach of [3] is to use a two-player game, that they call the "evolving tree game", which shares similarities with the "continuous tree-mining game" (CTM) presented in this paper. Contrarily to the aforementioned guarantees which are all of the form $\mathcal{O}(c(w)D)$, our bound in $\mathcal{O}\left(2L/k + kD\right)$ depends on the sum of all edge lengths L and on the depth D of the tree but not on its width w.

Notations and definitions. The following definitions are used throughout the paper. A *tree* $T = (V, E)$ is a connected acyclic graph. One specific node, called the root, is denoted $r \in V$. Every other node $u \in V \setminus \{r\}$ has a unique parent denoted by $p(u)$. For two nodes $u, v \in V$ we say that u is a descendant of v or equivalently that v is an ancestor of u and we denote by $u \preceq v$ if v can be obtained from u by iterating the parent function $p(\cdot)$. For any two nodes $u, v \in V$ we denote by $\mathrm{A}(u, v)$ the lowest common ancestor of u and v. We also denote by $u \to v$ (resp. $u \leftrightarrow v$) the sequence of nodes in the shortest path from u to v, excluding v (resp. including v). A leaf of T is a node $\ell \in V$ which has no descendant. The set of all leaves of T is denoted by $\mathcal{L}(T)$. We say that a tree is *simple* if no node has degree 2, except possibly for the root.

A *weighted* tree is a tree in which edges have a non-negative length. An unweighted tree can be seen as a weighted tree where all edge lengths are equal to 1. For $u \in V \setminus \{r\}$, the length of edge $(u, p(u))$ is denoted $d_u \in \mathbb{R}^+$. For any two nodes $u, v \in V$ we denote by $d(u, v)$ the distance from u to v, which can be defined by $d(u, v) = \sum_{w \in u \to \mathrm{A}(u,v)} d_w + \sum_{w \in v \to \mathrm{A}(u,v)} d_w$.

For some integer $k \geq 2$, a *discrete configuration* on a tree T is a collection $\boldsymbol{x} \in \mathbb{N}^{\mathcal{L}(T)}$ satisfying $\sum_{\ell \in \mathcal{L}(T)} x_\ell = k$. It can be extended to a collection $\boldsymbol{x} \in \mathbb{N}^V$ by setting $\forall u \in V : x_u = \sum_{\ell \preceq u} x_\ell$. The set of all discrete configurations is denoted $\mathcal{X}(T)$. A *fractional configuration* on T is a collection $\boldsymbol{y} \in \mathbb{R}_+^{\mathcal{L}(T)}$ satisfying $\sum_{\ell \in \mathcal{L}(T)} y_\ell = k$. It can be extended to a collection $\boldsymbol{y} \in \mathbb{R}_+^V$ by setting $\forall u \in V : y_u = \sum_{\ell \preceq u} y_\ell$. The set of all fractional configurations is denoted $\mathcal{Y}(T)$. For any two configurations (discrete or fractional) $\boldsymbol{x}$ and $\boldsymbol{x}'$, we define the optimal transport cost $\mathrm{OT}_T(\boldsymbol{x}, \boldsymbol{x}') = \sum_{u \in V} d_u |x_u - x'_u|$.

Structure of the paper. The paper is organized as follows. In Section 2 we define and analyze the continuous tree-mining game (CTM). In Section 3 we present the reductions that relating it to collective tree exploration (CTE) and layered tree traversal (LTT). In Section 4 we then apply the results of Section 2 to obtain new guarantees for both problems.

2 Analysis of the continuous tree-mining game

In this section, we introduce and analyze a two-player game that we call the continuous tree-mining game (CTM). We first define the game in Section 2.1, then we present an algorithm for the player of the game in Section 2.2, finally we provide an analysis of the player's algorithm in Section 2.3. The continuous tree-mining game is tightly connected to the problems of collective tree exploration (CTE) and of layered tree traversal (LTT), as we shall see in Section 3.

2.1 The continuous tree-mining game

The state of the game is defined at any continuous time t. It consists of a *simple weighted tree* $T(t)$, the evolution of which is controlled by the adversary, and of a *discrete configuration* $\boldsymbol{x}(t)$ over $T(t)$, the evolution of which is controlled by the player. We now precise the actions available to the player and the adversary at each instant.

Adversary. The adversary can do three things: kill a leaf, give some children to a leaf, or elongate the edge leading to a leaf. The first two operations occur instantaneously, while the other is performed continuously over time. We now detail all three operations.

Leaf edge elongation: Between discrete changes to the tree by the adversary, or discrete moves by the player, at any given continuous time t the adversary distinguishes one leaf $\ell(t)$ and lets the length $d_{\ell(t)}$ of the corresponding edge increase at unit rate. The adversary is only allowed to choose for $\ell(t)$ a leaf with more than one robot, i.e. such that $x_{\ell(t)}(t) \geq 2$.

Forking at a leaf: At discrete time points the adversary can choose some leaf ℓ hosting a number $x_\ell \geq 3$ of robots, and endow this leaf with some number $m \in \{2, \dots, x_\ell - 1\}$ of children. Denoting by $\ell_1, \dots, \ell_m$ these children, the newly created edges of the form $(\ell, \ell(i))$, $i \in [m]$, are initialised with some length δ, where $0 < \delta \leq 1$ is a quantity chosen by the player at the time of the fork.

Killing a leaf: At discrete time points the adversary can choose to kill some leaf ℓ. The corresponding edge $(\ell, p(\ell))$ is then also suppressed from the tree. In case node $p(\ell)$ is distinct from the root r, and had only one child u besides ℓ, we then merge the two edges $(u, p(\ell))$ and $(p(\ell), p(p(\ell)))$ into a single edge $(u, p(p(\ell)))$ (suppressing node $p(\ell)$) and endow this new edge with length $d_{p(\ell)} + d_u$, preserving the distance between $p(p(\ell))$ and u in the new tree. At all times, the tree thus remains *simple*.

Player. At any instant the player can move robots from one leaf to another one, by changing the configuration $\boldsymbol{x}$. The cost of going from configuration $\boldsymbol{x}$ to $\boldsymbol{x}'$ is equal to $\mathrm{OT}_T(\boldsymbol{x}, \boldsymbol{x}')$. When the adversary kills a leaf ℓ, the player is forced to move the corresponding x_ℓ robots to other leaves that are still alive and to pay the associated cost. When the adversary forks at some leaf ℓ, endowing it with m children, the player chooses the length $\delta < 1$ of the fork and must assign its x_ℓ robots to these newly created m leaves, paying the associated cost. Between two discrete re-allocations of robots, when the adversary elongates the leaf $\ell(t)$, the cost continuously increases at rate $x_{\ell(t)}$.

Goal of the player. A strategy for the player is a continuous-time algorithm which determines the response of the player to any modification of the tree by the adversary. It is formally defined as a function $\boldsymbol{x}' = \mathcal{A}(T, \boldsymbol{x}, T')$ which returns a new configuration $\boldsymbol{x}' \in \mathcal{X}(T')$ given the previous state of the game $\boldsymbol{x} \in \mathcal{X}(T)$ and a modified tree T'. Note that taking $T' = T$ allows to account for continuous elongation. We say that a strategy is $f(k, D)$-bounded, for some real-valued function $f(\cdot, \cdot)$, if it is such that no matter how the adversary plays, the cost incurred by the player is always less than $f(k, D)$, where D denotes the depth of the highest leaf. We will be particularly interested in the case where f is linear in D.

The interest of the continuous tree-mining game lies in the following reduction.

▶ **Theorem 1** (Propositions 12–16 in Section 3). *For any $f(k, D)$-bounded strategy for the continuous tree-mining game (CTM), there is a collective tree exploration (CTE) algorithm $\mathcal{A}$ such that for any unweighted tree T with n nodes and depth D,*

$$\texttt{Runtime}(\mathcal{A}, T) \leq \frac{2n + f(k, D)}{k} + D + 1.$$

Also, there exists a collection of layered tree traversal (LTT) randomized algorithms $\{\mathcal{A}_k\}_{k \in \mathbb{N}}$ satisfying for any layered tree T of length L and depth D,

$$\mathbb{E}(\texttt{Cost}(\mathcal{A}_k, T)) \leq \frac{2L + f(k, D)}{k} + 1.$$

The goal of the rest of this section is thus to prove the following theorem.

▶ **Theorem 2** (Proposition 11 in Section 2). *There exists a $f(k, D)$-bounded strategy for the continuous tree-mining game, with $f(k, D) = \mathcal{O}(k^2 D)$.*

2.2 A potential-based algorithm

We assign to any configuration $\boldsymbol{x} \in \mathcal{X}(T)$ a potential $\Psi(T, \boldsymbol{x})$ defined by

$$\Psi(T, \boldsymbol{x}) := \sum_{u \in V \setminus \{r\}} d_u \phi(x_u), \tag{1}$$

where ϕ is a strongly convex function to be determined later. We then define the strategy of the player by the following equation,

$$\mathcal{A}(T, \boldsymbol{x}, T') = \operatorname*{arg\,min}_{\boldsymbol{x}' \in \mathcal{X}(T')} \Psi(T', \boldsymbol{x}') + \mathrm{OT}_{T'}(\boldsymbol{x}, \boldsymbol{x}'), \tag{2}$$

in which ties are always broken arbitrarily in favor of any $\boldsymbol{x}' \neq \boldsymbol{x}$. Note that Algorithm (2) thus enforces the constraint that at all times t and for any configuration $\boldsymbol{x}' \neq \boldsymbol{x}(t)$, $\Psi(T(t), \boldsymbol{x}(t)) - \Psi(T(t), \boldsymbol{x}') < \mathrm{OT}_{T(t)}(\boldsymbol{x}(t), \boldsymbol{x}')$. We start by establishing some desirable properties on the dynamics of this algorithm.

▶ **Proposition 3** (Dynamics of $\boldsymbol{x}(t)$ following (2))**.** *While the adversary elongates a leaf, the moves of the player are all from the elongated leaf to other leaves of the tree, and no two robots are moved simultaneously. When the adversary deletes a leaf, all moves of the player are from the deleted leaf to other leaves. When the adversary forks a leaf l, with $m \leq x_l - 1$ children denoted $\{\ell_1, \dots, \ell_m\}$, there is a choice of a small real $\delta > 0$ such that the new configuration $\boldsymbol{x}'$ satisfies for all previously existing node $u \notin \{\ell_1, \dots, \ell_m\} : x_u = x'_u$ and for all newly created leaves $\ell \in \{\ell_1, \dots, \ell_m\} : x_\ell \in \{\lfloor x_l/m \rfloor, \lceil x_l/m \rceil\}$. At all times, the configuration of the game $\boldsymbol{x}$ satisfies $\forall \ell \in \mathcal{L}(T) : x_\ell \geq 1$.*

The proof of this proposition relies on the notion of "tension" between configurations. When the current tree T is clear from context, we will use $\Psi(\boldsymbol{x})$ as a shorthand for $\Psi(T, \boldsymbol{x})$. For any two configurations $\boldsymbol{x}$ and $\boldsymbol{x}'$ we call the tension from $\boldsymbol{x}$ to $\boldsymbol{x}'$ and we denote by $\tau(\boldsymbol{x} \to \boldsymbol{x}')$ the decrease in potential obtained when going from configuration $\boldsymbol{x}$ to configuration $\boldsymbol{x}'$, i.e. $\tau(\boldsymbol{x} \to \boldsymbol{x}') = \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}')$. For a configuration $\boldsymbol{x}$ and two leaves ℓ, ℓ' we call the tension from ℓ to ℓ' in $\boldsymbol{x}$ and denote by $\tau_{\boldsymbol{x}}(\ell \to \ell')$ the decrease in potential obtained by displacing a robot from ℓ to ℓ' in configuration $\boldsymbol{x}$, i.e. $\tau_{\boldsymbol{x}}(\ell \to \ell') = \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x} + \boldsymbol{e}_{\ell'} - \boldsymbol{e}_\ell)$. Note that this quantity is only defined if $x_\ell \geq 1$, which will always be the case as stated in Proposition 3. The structure imposed on the potential Ψ then leads to the following lemma, which essentially says that atomic moves (where only one robot moves at a time) are favored over simultaneous moves by Algorithm (2).

▶ **Lemma 4.** *Consider configurations $\boldsymbol{x}$ and $\boldsymbol{x}'$ such that $||\boldsymbol{x} - \boldsymbol{x}'||_1 = 2h$ (where $||\boldsymbol{x} - \boldsymbol{x}'||_1 = \sum_{\ell \in \mathcal{L}(T)} |x_\ell - y_\ell|$ is always even) and consider $\ell_1 \to \ell'_1, \dots, \ell_h \to \ell'_h$ an optimal transport plan going from $\boldsymbol{x}$ to $\boldsymbol{x}'$. The following inequality is always satisfied,*

$$\tau(\boldsymbol{x} \to \boldsymbol{x}') \leq \tau_{\boldsymbol{x}}(\ell_1 \to \ell'_1) + \dots + \tau_{\boldsymbol{x}}(\ell_h \to \ell'_h).$$

Furthermore, this inequality is strict if there are overlaps in the transport plan, i.e. a pair i, j such that the shortest paths $\ell_i \to \ell'_i$ and $\ell_j \to \ell'_j$ have an intersection of positive length.

Proof. We will show the property by induction on h. We observe that the property is true for $h = 1$. We now assume that the property is true for some $h \geq 1$ and aim to show it at $h + 1$. We consider two leaves $\ell \to \ell'$ of the transport plan from $\boldsymbol{x}$ to $\boldsymbol{x}'$ and we define $\boldsymbol{x}'' = \boldsymbol{x}' - \mathbf{z}$ with $\mathbf{z} = \boldsymbol{e}_{\ell'} - \boldsymbol{e}_\ell$, the configuration where the move $\ell \to \ell'$ did not take place. Observe that $||\boldsymbol{x} - \boldsymbol{x}''||_1 = 2h$ will later enable us to apply the induction hypothesis to $\tau(\boldsymbol{x} \to \boldsymbol{x}'')$. We decompose as follows,

$$\Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}') = \sum_{u \notin \ell \leftrightarrow \ell'} d_u(\phi(x_u) - \phi(x'_u)) + \sum_{u \in \ell \leftrightarrow \ell'} d_u(\phi(x_u) - \phi(x'_u)).$$

In the first sum we have $\forall u \notin \ell \leftrightarrow \ell' : x'_u = x''_u$ because the configuration value at these nodes is not affected by moves of the form $\ell \to \ell'$. In the second sum, we observe that by optimality of the transport map, the sign of $x'_u - x_u$ is always the same as the sign of z_u. This is because in an optimal transport plan, no two robots are transported on the same edge in opposite directions. Therefore we have the following inclusion on segments $[x_u + z_u, x'_u - z_u] \subset [x_u, x'_u]$, which by convexity of ϕ implies that the chord of the inner segment is below the chord of the outer segment, $\phi(x_u + z_u) + \phi(x'_u - z_u) \leq \phi(x_u) + \phi(x'_u)$. Rearranging the terms, we get the following identity, $\phi(x_u) - \phi(x'_u) \leq \phi(x_u) - \phi(x''_u) + \phi(x_u) - \phi(x_u + z_u)$. Combining these observations yields,

$$\Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}') \leq \sum_{u \in V} d_u(\phi(x_u) - \phi(x''_u)) + \sum_{u \in \ell \leftrightarrow \ell'} d_u(\phi(x_u) - \phi(x_u + z_u)),$$

where we recognize the first term to be $\tau(\boldsymbol{x} \to \boldsymbol{x}'')$ and the second term to be $\tau_{\boldsymbol{x}}(\ell \to \ell')$. We conclude by applying the induction hypothesis.

By strict convexity of ϕ, it is a direct observation that the inequality above is strict as soon as the inclusion $[x_u + z_u, x'_u - z_u] \subset [z_u, z'_u]$ is strict, i.e. when there are overlaps in the transport plan. ◀

The rest of the proof of Proposition 3 is decomposed into Lemmas 20–23, in Appendix A. These lemmas treat separately the cases of leaf elongation, leaf deletion and leaf fork, all of them relying on Lemma 4.

2.3 Competitive analysis by potential function

For a given tree T, we consider the optimal *fractional* configuration $\boldsymbol{y} = \{y_\ell\}_{\ell \in \mathcal{L}(T)} \in \mathcal{Y}(T)$ defined as the solution of the following convex optimization problem,

$$\begin{aligned} \min \quad & \Psi(T, \boldsymbol{y}) = \textstyle\sum_{u \in V \setminus \{r\}} d_u \phi(y_u) \\ \text{over} \quad & \boldsymbol{y} \in \mathcal{Y}(T), \end{aligned} \tag{3}$$

where we further assume that ϕ is twice differentiable on $\mathbb{R}^+$, with $\phi'' > 0$ and $\phi' > 0$.

The goal of this section is for some suitable constant γ, and some choice of function ϕ, to prove that the following equation is verified at all times, irrespective of the actions taken by the adversary,

$$\texttt{Cost}(t) + \Psi(T(t), \boldsymbol{x}(t)) \leq \gamma \Psi(T(t), \boldsymbol{y}(t)). \tag{4}$$

This readily leads to conclude that the strategy is $f(k, D)$-bounded, with $f(k, D) = \gamma\phi(k)D$. This is because $\gamma\phi(k)D$ is a simple upper bound of (3). The rest of this section is dedicated to showing (4) for a quadratic $\phi(\cdot)$ and a constant γ, thereby proving Theorem 2. We thus analyze the behaviour of $\boldsymbol{y}$, starting with Proposition 5 which provides a characterization of $\boldsymbol{y}$. We then study the dynamics of $\boldsymbol{y}$ over the course of the game (elongations, deletions, forks) in Proposition 6. After, we prove some relations between $\boldsymbol{x}$ and $\boldsymbol{y}$, for some quadratic function ϕ in Proposition 9. Finally, we conclude with the proof of (4) in Proposition 11.

▶ **Proposition 5** (Characterization of $\boldsymbol{y}$). *For any tree T of the game, Equation (3) has a unique solution. Furthermore, $\boldsymbol{y} \in \mathcal{Y}(T)$ is the solution of (3) if and only if there exists $\lambda \in \mathbb{R}$ s.t.*

$$\forall \ell \in \mathcal{L}(T), \sum_{u \in \ell \to r} d_u \phi'(y_u) \geq \lambda, \quad \textit{and} \quad y_\ell > 0 \Rightarrow \sum_{u \in \ell \to r} d_u \phi'(y_u) = \lambda. \tag{5}$$

Consequently, given any two leaves, ℓ, ℓ' for which $y_\ell, y'_\ell > 0$ we have

$$\sum_{u \in \ell \to A(\ell,\ell')} d_u \phi'(y_u) = \sum_{u \in \ell' \to A(\ell,\ell')} d_u \phi'(y_u). \tag{6}$$

Proof. We start by justifying the existence and uniqueness of $\boldsymbol{y}$. We note that for any tree T, the function $\boldsymbol{y} \to \Psi(T, \boldsymbol{y})$ is twice differentiable and compute its Hessian,

$$\nabla^2_{\boldsymbol{y}} \Psi(T, \boldsymbol{y}) = \left(\sum_{u \succeq \mathrm{A}(\ell,\ell')} d_u \phi''(y_u) \right)_{\ell,\ell' \in \mathcal{L}(T)} \tag{7}$$

which is an ultrametric matrix (see [8, Definition 3.2, p. 58]), because $\phi'' > 0$. We recall that an ultrametric matrix is always semidefinite positive because it can be seen as the covariance matrix of a Brownian motion on a tree (see e.g. [18]). Furthermore, it is non-singular if

no two rows are equal [8, Th 3.5 (ii)]. This applies here since $\forall u \in V : d_u > 0$. Thus $\boldsymbol{y} \to \Psi(T, \boldsymbol{y})$ is strictly convex and its minimum on the compact and convex set $\mathcal{Y}(T)$ exists and is unique.

We now give a characterization of $\boldsymbol{y}$ by applying the KKT conditions. Let $\mu = (\mu_\ell)_{\ell \in \mathcal{L}(T)}$ denote the vector of non-negative Kuhn-Tucker multipliers associated with inequality constraints $y_\ell \geq 0$, and $\lambda \in \mathbb{R}$ the multiplier associated with equality constraint $\sum_{\ell \in \mathcal{L}(T)} y_\ell = k$. We can characterize $\boldsymbol{y}$ by forming the Lagrangian

$$L(\mathbf{z}; (\lambda, \mu)) := \sum_{u \in V(T) \setminus \{r\}} d_u \phi(z_u) - \sum_{\ell \in \mathcal{L}(T)} \mu_\ell z_\ell + \lambda \left[k - \sum_{\ell \in \mathcal{L}(T)} z_\ell \right].$$

Optimality of $\boldsymbol{y}$ is characterized by the existence multipliers λ, μ such that the stationarity conditions together with complementary conditions are satisfied,

$$\forall \ell \in \mathcal{L}(T), \sum_{u \in \ell \to r} d_u \phi'(y_u) - \mu_\ell - \lambda = 0, \quad \text{and} \quad \forall \ell \in \mathcal{L}(T),\ \mu_\ell y_\ell = 0.$$

This is readily seen to be equivalent to the conditions stated in the Lemma. ◀

▶ **Proposition 6** (Dynamics of $\boldsymbol{y}(t)$ following (3))**.** *Consider some time interval $I = [t_1, t_2]$, in which all leaves ℓ satisfy $y_\ell > 0$ and for which the only edge length to increase is d_l, then y_l decreases and for any leaf $\ell \neq l$, y_ℓ increases. Furthermore, $\boldsymbol{y}$ is differentiable and its derivative $\dot{\boldsymbol{y}}$ satisfies*

$$\dot{y}_l \geq -\frac{1}{d_l} \frac{\max \phi'}{\min \phi''}. \tag{8}$$

Consider a leaf l that undergoes a discrete step (fork or deletion), then the optimal configuration y_ℓ of any leaf $\ell \neq l$ is increased. Furthermore, if l undergoes a fork of length δ with m children, one has

$$y'_l - y_l \geq -\frac{\delta}{d_l} \frac{\max \phi'}{\min \phi''}, \tag{9}$$

where $\boldsymbol{y}'$ denotes the optimal configuration right after the fork took place, in the tree in which l is now the parent of m children at distance δ and all attributed configuration weight y'_l/m.

▶ Remark 7. The assumption that $y_\ell > 0$ is not required in practice, but it simplifies the argument and it is always verified for the given Algorithm (2) of the player, as we shall later see.

Proof. We treat separately the cases of elongation, deletion and fork.

Leaf elongation. Consider the leaf l which is being elongated during interval $[t_1, t_2]$, all other edges being left unchanged. Consider some $t \in [t_1, t_2)$ satisfying $\boldsymbol{y}(t) > 0$ and some small $dt > 0$. The differentiability of $\boldsymbol{y}(\cdot)$ at t can be deduced from straightforward arguments using the inverse function theorem and the smooth evolution of the convex potential. We thus focus here on the sign of derivatives $\dot{\boldsymbol{y}}(t)$. We have by the characterization of $\boldsymbol{y}(t + dt)$ and of $\boldsymbol{y}(t)$ given in (5) that $\nabla_{\boldsymbol{y}} \Psi(T(t + dt), \boldsymbol{y}(t + dt)) - \nabla_{\boldsymbol{y}} \Psi(T(t), \boldsymbol{y}(t)) \in \mathrm{Vect}(\mathbb{1})$. Using the identity $\nabla_{\boldsymbol{y}} \Psi(T(t + dt), \boldsymbol{y}(t + dt)) = \nabla_{\boldsymbol{y}} \Psi(T(t), \boldsymbol{y}(t + dt)) + \phi'(y_l(t + dt)) \boldsymbol{e}_l dt$ and a Taylor expansion, we obtain,

$$\nabla^2_{\boldsymbol{y}} \Psi(T(t), \boldsymbol{y}(t))(\boldsymbol{y}(t + dt) - \boldsymbol{y}(t)) + \phi'(y_l(t + dt)) \boldsymbol{e}_l dt + o(dt) \in \mathrm{Vect}(\mathbb{1}),$$

which in the limit $dt \to 0$ gives,

$$U(t)\dot{\boldsymbol{y}}(t) \in -\phi'(y_l(t))\boldsymbol{e}_l + \mathrm{Vect}(\mathbb{1}) \tag{10}$$

where for shorthand we denote $U(t) = \nabla^2_{\boldsymbol{y}}\Psi(T(t), \boldsymbol{y}(t))$, the ultra-metric defined in (7). Lemma 8 below applied to (10) then allows to conclude that $\dot{y}_l < 0$ and $\dot{y}_\ell \geq 0$ for all $\ell \neq l$.

▶ **Lemma 8.** *Assume $U \in \mathbb{R}^{n\times n}$ a positive definite ultrametric matrix satisfies*

$$U\mathbf{z} = -\mu\boldsymbol{e}_l + \lambda\mathbb{1} \tag{11}$$

for constants $\lambda \in \mathbb{R}$, $\mu > 0$, for some index $l \in [n]$ and some zero sum vector $\mathbf{z} \in \mathbb{R}^n$, i.e. such that $\mathbf{z}^T\mathbb{1} = 0$. It is then the case that $\lambda \geq 0$ and that $\forall \ell \in [n] \setminus \{l\} : z_\ell \geq 0$.

Proof. We denote by M the inverse of the positive-definite ultrametric U, i.e. $M = U^{-1}$, which satisfies the following properties, [8, Th. 3.5 (i)] (a) the diagonal elements of M are non-negative (b) the off-diagonal elements of M are non-positive (c) the sum of elements of M across a row (or column) is non-negative.

By multiplying (11) on the left by $\mathbb{1}^T M$ and using that $\mathbb{1}^T\mathbf{z} = 0$, we get $\mu\mathbb{1}^T M\boldsymbol{e}_l = \lambda\mathbb{1}^T M\mathbb{1}$. By property (c) above, $\mathbb{1}^T M\boldsymbol{e}_l \geq 0$ and since $\mu \geq 0$ we get $\lambda \geq 0$. We then have $z = -\mu Me_l + \lambda M\mathbb{1}$, thus for $\ell \neq l$ we get $z_\ell = -\mu\boldsymbol{e}_\ell^T M\boldsymbol{e}_l + \lambda\boldsymbol{e}_\ell^T M\mathbb{1}$. Since $\boldsymbol{e}_\ell^T M\boldsymbol{e}_l \leq 0$ by property (b) and $\boldsymbol{e}_\ell^T M\mathbb{1} \geq 0$ by property (c) we get $z_\ell \geq 0$. Finally, observe that since $\mathbb{1}^T\mathbf{z} = 0$, it must be the case that $z_l \leq 0$. ◀

We now consider a leaf $\ell \neq l$ that is a descendant of $p(l)$. Using equation (6) we get $\sum_{u\in\ell\to p(l)} d_u\phi'(y_u) = d_l\phi'(y_l)$ at all times. Taking time derivatives, we obtain,

$$\sum_{u\in\ell\to p(l)} d_u\dot{y}_u\phi''(y_u) = \phi'(y_l) + d_l\dot{y}_l\phi''(y_l).$$

We have shown that the left-hand side is non-negative, thus ensuring,

$$\dot{y}_l \geq -\frac{1}{d_l}\frac{\phi'(y_l)}{\phi''(y_l)} \geq -\frac{1}{d_l}\frac{\max\phi'}{\min\phi''}.$$

Leaf fork. We now consider the case when leaf l undergoes a discrete fork with m children, each attached to an edge of length δ. This case will be treated similarly as that of a continuous elongation by introducing $s \in [0,1]$ and $T(s)$ the tree constructed from T by providing l with m children each attached to a leaf of length $s\delta$. It is clear that $T(0)$ corresponds to the tree before the fork and that $T(1)$ corresponds to the tree after the fork. We shall show that optimal configuration $\boldsymbol{y}(s)$ of $T(s)$ satisfies the property that $y_\ell(s)$ is increasing if ℓ is not a child of l. It is clear from the strict convexity of $\boldsymbol{y} \to \Psi(T(s), \boldsymbol{y})$ and by symmetry that the value of $y_\ell(s)$ for all children ℓ of l is equal to $\frac{1}{m}y_l(s)$, as soon as $s > 0$. We will thus see $\boldsymbol{y}(s)$ as a configuration of the tree before the fork, but for the modified potential $\Psi(s, \boldsymbol{y}) = \Psi(T(0), \boldsymbol{y}) + s\delta \times m\phi\left(\frac{1}{m}y_l\right)$. By the same reasoning as above, we get the following dynamics

$$U(s)\dot{\boldsymbol{y}}(s) \in -\delta\phi'\left(y_l(s)/m\right)\boldsymbol{e}_l + \mathrm{Vect}(\mathbb{1}),$$

where $U(s) = \left(\sum_{u\succeq \mathrm{A}(\ell,\ell')} d_u\phi''(y_u(s))\right)_{\ell,\ell'\in\mathcal{L}(T)} + \left(\mathbb{1}(\ell = \ell' = l)\frac{s\delta}{m}\phi''(y_l(s))\right)_{\ell,\ell'\in\mathcal{L}(T)}$ is a positive definite ultra-metric to which we apply Lemma 8 to get the desired monotonicity. Then by considering a leaf $\ell \neq l$ that is a descendant of $p(l)$, we have $\sum_{u\in\ell\to p(l)} d_u\phi'(y_u(s)) = d_l\phi'(y_l(s)) + \delta s\phi'\left(y_l(s)/m\right)$. By taking derivatives on s, the quantity, $d_l\phi''(y_l(s))\dot{y}_l(s) + \frac{s\delta}{m}\phi''\left(y_l(s)/m\right)\dot{y}_l(s) + \delta\phi'\left(y_l(s)/m\right)$ is non-negative, ensuring that $\dot{y}_l(s) \geq -\frac{\delta}{d_l}\frac{\phi'(y_l(s)/m)}{\phi''(y_l(s))}$, and thus $y_l' - y_l \geq -\frac{\delta}{d_l}\frac{\max\phi'}{\min\phi''}$.

Leaf deletion. Finally, we consider the case of the deletion of leaf l. We observe as in [3], that the configuration after the deletion of a leaf can be obtained as the limit configuration when this leaf is extended to infinity. Such infinite extension only leads to increasing the value of the configuration at all other leaves, thereby proving the monotonicity statement made for deletions. ◀

▶ **Proposition 9** (Bounds on $\boldsymbol{x}$ and $\boldsymbol{y}$). *Let $\epsilon, \epsilon' \in (0, 1/2]$ be two fixed constants, and assume that $\phi(x) = ax + bx^2$ where parameters a, b are chosen such that*

$$2bk \leq \epsilon' a \text{ and } b(2 - 2\epsilon - \epsilon') \geq 2 + \epsilon'. \tag{12}$$

Then Algorithm (2) *is such that irrespective of the adversary's moves, at all times,*

$$\forall \ell \in \mathcal{L}(T),\ x_\ell < y_\ell + 2 - \epsilon, \tag{13}$$

and,

$$\forall \ell \in \mathcal{L}(T),\ \epsilon \leq y_\ell. \tag{14}$$

Proof. We start by showing how (14) follows from (13). Recall from Proposition 6 that the value of y_ℓ can only decrease if ℓ is elongated or forked. Since a leaf can only be elongated if $x_\ell \geq 2$, it follows from (13) that y_ℓ can not go below ϵ during an elongation. Now assume that the leaf ℓ undergoes a fork with $m \leq x_\ell - 1$ children. By (13), it must be the case that $y_l \geq m - 1 + \epsilon \geq m\epsilon + (1 - \epsilon)$. For a small enough value of δ in equation (9), we get that $y'_l/m > \epsilon$, just after the fork. Thus neither elongation nor a fork may make a value of $\boldsymbol{y}$ go beneath ϵ.

We now show how (13) follows from (12). Note first that (13) holds true at $t = 0$. Consider then the earliest time at which this inequality is met with equality: there is some leaf ℓ such that $x_\ell = y_\ell + 2 - \epsilon$. Then, since $x_r = y_r = k$, there exists an ancestor v of ℓ such that $x_v < y_v + 2 - \epsilon$ but $x_u \geq y_u + 2 - \epsilon$ for all $u \in \ell \to v$. Also, since $x_v = \sum_{u:p(u)=v} x_u$ and $y_v = \sum_{u:p(u)=v} y_u$ there must be a child w of v such that $x_w < y_w$. By iterating this argument until we reach a leaf, there must exist $\ell' \in \mathcal{L}(T)$ descending from v and satisfying that for all $u \in \ell' \to v : x_u < y_u$. Let $\boldsymbol{x}'$ be the robot configuration obtained from $\boldsymbol{x}$ by moving one robot from ℓ to ℓ'. We evaluate the difference

$$\Delta := \Psi(T, \boldsymbol{x}') + \mathrm{OT}_T(\boldsymbol{x}, \boldsymbol{x}') - \Psi(T, \boldsymbol{x}).$$

This reads

$$\begin{aligned}
\Delta &= \sum_{u \in \ell \to v} d_{i(j)}[1 - a + b\{(x_u - 1)^2 - x_u^2\}] + \sum_{u \in \ell' \to v} d_u[1 + a + b\{(x_u + 1)^2 - x_u^2\}] \\
&= \sum_{u \in \ell \to v} d_u[1 - a + b(1 - 2x_u)] + \sum_{\ell' \to v} d_u[1 + a + b(1 + 2x_u)] \\
&\leq \textstyle\sum_{u \in \ell \to v} d_{i(j)}[1 - a + b(1 - 2(y_u + 2 - \epsilon))] + \sum_{\ell' \to v} d_{i'(j)}[1 + a + b(1 + 2y_u)],
\end{aligned}$$

where we used the inequalities between $\boldsymbol{x}$ and $\boldsymbol{y}$.

We then recall from (6) (noting that $\phi'(x) = a + 2bx$):

$$\sum_{u \in \ell \to v} d_u[a + 2by_u] = \sum_{u \in \ell' \to v} d_u[a + 2by_u]. \tag{15}$$

We can thus simplify the previous bound on Δ to obtain

$$\Delta \leq \sum_{u \in \ell \to v} d_u[1 + b(1 - 2(2 - \epsilon))] + \sum_{u \in \ell' \to v} d_u[1 + b]. \tag{16}$$

Equation (15) together with the assumption $2bk \leq \epsilon' a$ made in the proposition entails, noticing that $y_v \leq k$ for all v, that

$$\sum_{u \in \ell \to v} d_u \leq (1+\epsilon') \sum_{u \in \ell \to v} d_u, \quad \text{and similarly,} \quad \sum_{u \in \ell' \to v} d_u \leq (1+\epsilon') \sum_{u \in \ell \to v} d_u. \tag{17}$$

In view of (16), we thus have

$$\Delta \leq \sum_{u \in \ell \to v} d_u \left[1 + b(-3+2\epsilon) + (1+\epsilon')(1+b)\right] = \sum_{u \in \ell \to v} d_u \left[2 + \epsilon' - b(2 - 2\epsilon - \epsilon')\right].$$

This upper bound is negative under the second condition $b(2-2\epsilon-\epsilon') \geq 2+\epsilon'$ of the proposition, which is impossible by Algorithm (2). ◀

▶ Remark 10. We will choose as specific values for $\phi(x) = ax + bx^2$ the quantities $a = 20k$, and $b = 5$, as well as $\epsilon = \epsilon' = 1/2$. These quantities satisfy the conditions of Proposition 9.

▶ **Proposition 11.** *The above-defined strategy, for the parameters in Remark 10, is such that at all times, the configuration $(T(t), \boldsymbol{x}(t))$ of the game satisfies* (4) *for $\gamma = 48$.*

$$\texttt{Cost}(t) + \Psi(T(t), \boldsymbol{x}(t)) \leq \gamma \Psi(T(t), \boldsymbol{y}(t)). \tag{4}$$

Proof. We show (4) is preserved by leaf elongations, leaf deletions and leaf forks.

Leaf elongation. We first consider the evolution of the equation during a leaf elongation and between instants at which the player re-assigns robots. The three quantities involved in (4) evolve smoothly. Denoting by $\ell(t)$ the leaf chosen by the adversary for elongation, we have:

$$\begin{aligned}
&\tfrac{d}{dt}\texttt{Cost}(t) = x_{\ell(t)}(t),\\
&\tfrac{d}{dt}\Psi(T(t), \boldsymbol{x}(t)) = \phi(x_{\ell(t)}(t)),\\
&\tfrac{d}{dt}\Psi(T(t), \boldsymbol{y}(t)) = \phi(y_{\ell(t)}(t)),
\end{aligned}$$

where for the third identity we used the optimality of $\boldsymbol{y}(t)$.

We thus want to choose γ such that, writing $x = x_{\ell(t)}(t)$ and $y = y_{\ell(t)}(t)$ for short, the following inequality holds:

$$x + \phi(x) \leq \gamma \phi(y). \tag{18}$$

Proposition 9 gives us the following relations:

$$x \leq y + 2 - \epsilon = y + 3/2 \quad \text{and} \quad y \geq \epsilon = 1/2.$$

We then notice that $\forall y \geq 1/2 : y + 3/2 + \phi(y + 3/2) \leq \gamma\phi(y)$ is satisfied for any $\gamma \geq 7$, and for the choice of parameters made in Remark 10, therefore proving (18).

Next, we consider an instant t, occurring during a leaf elongation, at which the player reassigns robots, moving from $\boldsymbol{x}$ to $\boldsymbol{x}'$. The cost incurred by the player for this move is precisely $\mathrm{OT}_{T(t)}(\boldsymbol{x}, \boldsymbol{x}')$. Because of the player's strategy, this move occurs only if

$$\mathrm{OT}_{T(t)}(\boldsymbol{x}, \boldsymbol{x}') + \Psi(T(t), \boldsymbol{x}') \leq \Psi(T(t), \boldsymbol{x}).$$

Thus the left-hand side of (4) makes a downward jump at time t, whereas its right-hand side incurs no jump.

Leaf fork. The case of leaf fork is similar to the case of leaf elongation. Denote by T, $\boldsymbol{x}$ and $\boldsymbol{y}$ (resp T', $\boldsymbol{x}'$ and $\boldsymbol{y}'$) the tree, the discrete configuration and the optimal configurations before (resp. after) the fork and denote by ℓ the leaf that is forked. Recall that the fork length δ is chosen small enough for the fork to lead to no reassignments, i.e. $x_\ell = x'_\ell$ and the newly created leaves take their values in $\{\lceil x_\ell/m \rceil, \lfloor x_\ell/m \rfloor\}$. The fork leads to the following change in cost and potential,

$$\begin{aligned}
&\Delta_{\boldsymbol{x}}\texttt{Cost} = \delta x_\ell,\\
&\Delta_{\boldsymbol{x}}\Psi = \Psi(T', \boldsymbol{x}') - \Psi(T, \boldsymbol{x}) \leq m\delta\phi(\lceil x_\ell/m \rceil),\\
&\Delta_{\boldsymbol{y}}\Psi = \Psi(T', \boldsymbol{y}') - \Psi(T, \boldsymbol{y}) \geq \textstyle\int_0^\delta m\phi(y_\ell(s)/m)ds \geq m\delta\phi(y'_\ell/m),
\end{aligned}$$

where for the third inequality, we used the fact that a leaf fork can be seen as a leaf elongation with modified leaf potential : $y \to m\phi(y/m)$, see Proposition 6. Further by Proposition 6 the value of $\delta > 0$ can be chosen small enough to ensure $y'_\ell/m \geq \epsilon$ and $y'_\ell \geq y_\ell - 1/2$. We then use $x_\ell \leq y_\ell + 3/2 \leq y'_\ell + 2$ to observe that $\Delta_{\boldsymbol{x}}\texttt{Cost} \leq \Delta_{\boldsymbol{y}}\Psi$, and use $\lceil x_\ell/m \rceil \leq y'_\ell/m + 3/2$ and $y'_\ell/m \geq \epsilon$ along with the computations used for leaf elongations to get that $\Delta_{\boldsymbol{x}}\Psi \leq \gamma\Delta_{\boldsymbol{y}}\Psi'$ for any $\gamma \geq 7$. Summing up both inequalities, we get that for any $\gamma \geq 8$,

$$\Delta_{\boldsymbol{x}}\texttt{Cost} + \Delta_{\boldsymbol{x}}\Psi \leq \gamma\Delta_{\boldsymbol{y}}\Psi'.$$

Leaf deletion. We finally deal with leaf deletions and keep the same notations of $(T, \boldsymbol{x}, \boldsymbol{y})$ and $(T', \boldsymbol{x}', \boldsymbol{y}')$ to denote the state of the game before and after the deletion and use ℓ to denote the deleted leaf. We first lower-bound the value of $\Delta_{\boldsymbol{y}}\Psi = \Psi(T', \boldsymbol{y}') - \Psi(T, \boldsymbol{y})$ and then upper bound the value of $\Delta_{\boldsymbol{x}}(\Psi + \texttt{Cost}) = \Psi(T', \boldsymbol{x}') - \Psi(T, \boldsymbol{x}) + \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}')$.

We define $\boldsymbol{\delta} = \boldsymbol{y}' - \boldsymbol{y} \in \mathcal{Y}(T)$ where we set by convention $y'_\ell = 0$. For each $s \in [0, 1]$, we then let $\boldsymbol{y}(s) = \boldsymbol{y} + s\boldsymbol{\delta} \in \mathcal{Y}(T)$. We then define $h(s) := \sum_u d_u\phi(y_u(s))$, so that

$$h(0) = \Psi(T, \boldsymbol{y}),\ h(1) = \Psi(T', \boldsymbol{y}').$$

By optimality of $\boldsymbol{y}$ for T, $h'(0) = 0$. Also,

$$h''(s) = \sum_u d_u\delta_u^2(2b) \geq d_\ell(y_\ell)^2(2b).$$

Thus $h'(s) \geq 2bs * d_\ell(y_\ell)^2$. In turn one obtains

$$\Delta_{\boldsymbol{y}}\Psi = \Psi(T', \boldsymbol{y}') - \Psi(T, \boldsymbol{y}) = h(1) - h(0) \geq bd_\ell(y_\ell)^2. \tag{19}$$

Let us now upper-bound the increase to the left-hand side of (4) incurred by the player's move. Note that the player's strategy is to choose a new configuration $\boldsymbol{x}' \in \mathcal{Y}(T')$ that minimizes $\mathrm{OT}_T(\boldsymbol{x}, \boldsymbol{x}') + \Psi(\boldsymbol{x}') - \Psi(\boldsymbol{x})$, which is precisely the amount by which this left-hand side will increase. We can thus choose any target configuration $\boldsymbol{x}'' \in \mathcal{Y}(T')$ we like to upper-bound this left-hand side increase. Pick one leaf $\ell' \neq \ell$ that is a descendant of $p(\ell)$, and consider the configuration $\boldsymbol{x}'' = \boldsymbol{x} - x_\ell\boldsymbol{e}_\ell + x_\ell\boldsymbol{e}_{\ell'}$ obtained by moving the x_ℓ robots from ℓ to ℓ'. Using (17), we observe that the associated transport cost $\mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}'')$ is no larger than $(5/2)d_\ell x_\ell$. The difference in potential then writes as follows

$$\Psi(\boldsymbol{x}'') - \Psi(\boldsymbol{x}) = \sum_{u \in \ell \to \mathrm{A}(\ell,\ell')} d_u(\phi(x_u - x_\ell) - \phi(x_u)) + \sum_{u \in \ell' \to \mathrm{A}(\ell,\ell')} d_u(\phi(x_u + x_\ell) - \phi(x_u)).$$

We then observe that for $\phi(x) = ax + bx^2$, we have for $x, h \geq 0$,

$$\begin{aligned}
&\phi(x + h) - \phi(x) = bh^2 + (a + 2bx)h \leq bh^2 + (\phi(x + 1) - \phi(x))h\\
&\phi(x - h) - \phi(x) = bh^2 - (a + 2bx)h \leq bh^2 + (\phi(x - 1) - \phi(x))h
\end{aligned}$$

and thus, we bound $\Psi(\boldsymbol{x}'') - \Psi(\boldsymbol{x})$ by,

$$\begin{aligned} & bx_\ell^2 d(\ell,\ell') + \sum_{u\in\ell\to \mathrm{A}(\ell,\ell')} x_\ell d_u(\phi(x_u-1)-\phi(x_u)) + \sum_{u\in\ell'\to \mathrm{A}(\ell,\ell')} x_\ell d_u(\phi(x_u+1)-\phi(x_u)) \\ & \qquad \le bx_\ell^2 d(\ell,\ell') + x_\ell \tau_{\boldsymbol{x}}(\ell'\to\ell) \\ & \qquad \le bx_\ell^2 d(\ell,\ell') + x_\ell d(\ell,\ell') \end{aligned}$$

where in the last equation we used the inequality $\tau_x(\ell\to\ell') \le d(\ell,\ell')$, which was proved in Proposition 3. This yields the following bound,

$$\Psi(\boldsymbol{x}'') - \Psi(\boldsymbol{x}) \le bx_\ell^2 d(\ell,\ell') + x_\ell d(\ell,\ell') \le d(\ell,\ell')(y_\ell)^2(16b+8),$$

where we used $x_\ell \le 4y_\ell$ which follows from (13) and $y_\ell \ge 1/2$. Adding the transport cost of $\mathrm{OT}(\boldsymbol{x}'',\boldsymbol{x}) = x_\ell d(\ell,\ell')$ we get,

$$\Delta_{\boldsymbol{x}}(\Psi + \texttt{Cost}) \le (y_\ell)^2 d(\ell,\ell')(16b+16) \le (y_\ell)^2 d_\ell(5/2)(16b+16) \le \gamma \Delta_{\boldsymbol{y}}\Psi$$

for $\gamma = (5/2)(16b+16)/b = 48$ where we used the lower bound of $\Delta_{\boldsymbol{y}}\Psi$ given by (19). ◀

3 Reductions of Continuous Tree-Mining Game

In this section, we precise the reductions that connect the game above to both of our motivations, collective tree exploration (CTE) and layered tree traversal (LTT). The reductions are summarized in the following diagram, Figure 1.

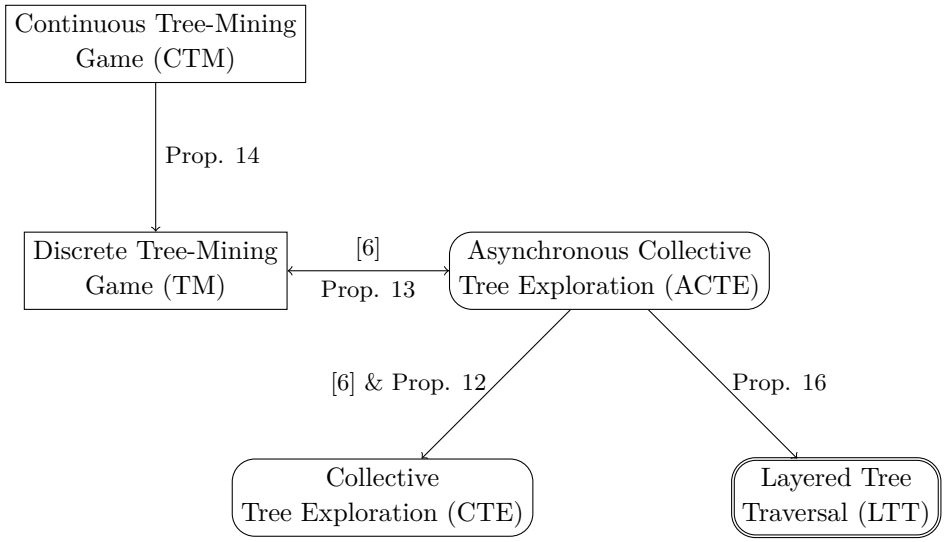

Figure 1 Summary of reductions. Rectangles are for settings expressed in the form of a game. Ovals are for exploration problems. Double boundary are for online problems.

3.1 Synchronous and Asynchronous Collective Tree Exploration

The problem of (synchronous) collective tree exploration (CTE) was introduced by [14]. We shall first recall the definition of the problem and then present an extension called asynchronous collective tree exploration (ACTE) [6]. We finally recall the reduction from (CTE) to (ACTE) in Proposition 12.

(Synchronous) Collective Tree Exploration (CTE). The problem is defined as follows. A team of k robots is initially located at the root of an unknown tree T with edges of length 1. The team is tasked to go through the edges of a tree as fast as possible and then return to the root. At all rounds, the robots can move synchronously along one incident edge, thereby possibly visiting new nodes. When a robot visits a node u for the first time, the full team becomes aware of its degree and of all the unexplored edges adjacent to u, that we shall call *dangling* edges. When there are no more dangling edges, the entire team knows that all edges have been explored and meets at the root to declare the termination of exploration. The runtime of an exploration algorithm $\mathcal{A}$ is defined on a tree T as the number of rounds before termination and is denoted by $\texttt{Runtime}(\mathcal{A}, k, T)$. By extension, for any two integers $n \geq D$, we denote by $\texttt{Runtime}(\mathcal{A}, k, n, D)$ the maximum number of rounds required before termination on any tree with n nodes and depth D. For more formalism, we refer to [7].

Asynchronous Collective Tree Exploration (ACTE). The problem of asynchronous collective tree exploration (ACTE) is a generalization of collective tree exploration (CTE) in which the agents move sequentially. At each (discrete) time t, only a single robot indexed by $r_t \in [k]$ is allowed to move. The sequence $r_0, r_1, \dots$ is completely adversarial. At the beginning of move t, the team is *only* given the information of whether robot r_t is adjacent to some unexplored edge, and if so, the ability to move along that edge. Consequently in contrast with collective tree exploration, the team does not know the degree of a node until it has been *mined*, i.e. some robot moving from that node was not given a fresh unexplored edge to traverse. Exploration starts with all robots located at the root of some unknown tree T and ends only when all nodes have been mined. We do not ask that all robots return to the root at the end of exploration for this might not be possible in finite time if say some robot is allowed only a single move. For an asynchronous collective tree exploration algorithm $\mathcal{B}$, we denote by $\texttt{Moves}(\mathcal{B}, k, T)$ the maximum number of moves that are needed to traverse all edges of T. We naturally extend this notation to $\texttt{Moves}(\mathcal{B}, k, n, D)$ for two integers $n \geq D$ denoting the maximum number of moves required to explore a tree with n nodes and depth D. (ACTE) generalises (CTE) in the following sense.

▶ **Proposition 12.** *For any asynchronous exploration collective tree exploration algorithm $\mathcal{B}$, one can derive a synchronous algorithm $\mathcal{A}$ satisfying,*

$$\texttt{Runtime}(\mathcal{A}, k, T) \leq \left\lceil \frac{1}{k} \texttt{Moves}(\mathcal{B}, k, T) \right\rceil + D,$$

for any tree T of depth D, and for all number of robots k.

Proof idea. Considering a sequence of allowed moves $1, 2, \dots, k, 1, 2, \dots, k, 1, 2, \dots$, we recover a synchronous collective exploration algorithm from an asynchronous collective exploration algorithm. Then all robots return to the root in at most D additional steps. We refer to [6] for more details. ◀

3.2 Continuous and Discrete Tree-Mining Games

The problem of asynchronous collective tree exploration can be reduced to a game opposing a player and an adversary known as the "tree-mining" game (TM) [6]. In this section, we first recall the original setting of the discrete tree-mining game (TM), and then show it can be reduced to the continuous tree-mining game (CTM), which was studied in Section 2.

Discrete Tree-Mining Game (TM). The game is defined as follows for some fixed integer $k \geq 2$. At step $i \geq 0$, the state of the game is defined by a pair $\mathcal{T}(i) = (T(i), \boldsymbol{x}(i))$ where $T(i) = (V(i), E(i), L(i))$ is a rooted tree with nodes $V(i)$, edges $E(i)$, and with $L(i) \subset \mathcal{L}(T(i))$ a subset of the leaves of $T(i)$ called *active* leaves. The configuration $\boldsymbol{x}(i) = (x_\ell(i))_{\ell \in L(i)}$ represents the number of miners on each active leaf, for a total of k miners, i.e. $\sum_{\ell \in L(i)} x_\ell(i) = k$. At the beginning of step $i \in \mathbb{N}$, the adversary is the first to play. It chooses an active leaf $\ell(i) \in L(i)$ that becomes inactive and is given $x_{\ell(i)}(i) - 1$ (active) children which are added to $L(i+1)$. Then, the player sends one miner of $\ell(i)$ to each of newly created leaves, and decides on some active leaf $\ell'(i) \in L(i+1)$ on which to send the last miner at $\ell(i)$. After round i, the cost of the game is updated as follows,

$$\mathrm{Cost}(i+1) = \mathrm{Cost}(i) + d(\ell(i), \ell'(i)),$$

keeping track of the total distance traversed by the excess miners[1]. A strategy for the player of the tree-mining game is called $f(k, D)$-bounded if all k miners are guaranteed to attain depth D with a score of at most $f(k, D)$. We have the following result,

▶ **Proposition 13.** *For every $f(k, D)$-bounded strategy for the tree-mining game (TM) there exists an asynchronous collective tree exploration (ACTE) algorithm $\mathcal{B}$ satisfying,*

$$\texttt{Moves}(\mathcal{B}, k, n, D) \leq 2n + f(k, D).$$

Proof idea. The proof relies on the notion of locally-greedy algorithms with target, formally defined in [6]. An exploration algorithm is called locally-greedy if moving robots always prefer unexplored edges over explored edges. In a locally-greedy algorithm with targets, each robot is assigned a target, which is a discovered node that has not been mined. When a robot is not adjacent to an unexplored edge, it simply performs one step towards its target. In the reduction to the tree-mining game, the targets correspond to the active leaves. When a robot mines its target, all robots sharing the same target must be re-targeted, and the additional movement cost associated to this re-targeting is bounded by the distance between the previous target and the new target. This quantity is exactly the cost in the corresponding tree-mining game. We refer to [6] for more details. ◀

▶ Remark. In the description of the game above, leaves are not deleted but rather deactivated, and some nodes can be of degree 2. An alternative definition of the game, which is closer from the one used in the continuous tree-mining game, would see the deactivated leaves effectively deleted from the tree, as well as nodes of degree equal to 2 by merging their adjacent edges. This view, which is used for the reduction below, maintains a "simple" tree, as defined in the introduction.

Continuous Tree-Mining Game (CTM). We defined in the beginning of Section 2 a variant of the tree-mining game, called the continuous tree-mining game (CTM). Similarly, we say that a strategy for this variant of the game is $f(k, D)$-bounded if k miners are guaranteed to attain depth D with a cost of at most $f(k, D)$. We shall now prove a reduction from the discrete tree-mining game to the continuous tree-mining game.

▶ **Proposition 14.** *For any $f(k, D)$-bounded strategy for the player of the continuous game (CTM), there is an $f(k, D)$-bounded strategy for the player of the discrete tree game (TM).*

[1] This cost can slightly be refined, see [6].

Proof. The discrete strategy is defined using the continuous strategy as follows. Given the i-th move of the adversary, the player of the discrete game emulates the continuous game for some duration $t_{i+1} - t_i$ against some adequately defined continuous adversary. The configuration obtained in the continuous tree T^c at time t_{i+1} defines the response of the player on the discrete tree T^d at discrete step i. The following properties, which highlight the correspondences between the continuous and the discrete games will always be satisfied at instants $t_1, t_2, t_3, \dots$ (a) $T^c(t_i)$ and $T^d(i)$ have the same set of vertices V, edges E and leaves $\mathcal{L}$. (b) For any leaf $\ell \in \mathcal{L} : x^c_\ell = x^d_\ell$. (c) For any $v \in V \setminus \mathcal{L} : d^c_v = d^d_v$ where d^c_v (resp. d^d_v) denotes the distance from v to $p(v)$ in T^c (resp. T^d). (d) If for some leaf $\ell \in \mathcal{L}$ we have $d^c_\ell < d^d_\ell$ then necessarily, $x^c_\ell = x^d_\ell = 1$. We assume that the following properties are satisfied at time t_i and show how we define time t_{i+1} as well as the evolution of the continuous game during $[t_i, t_{i+1}]$. The state of $T^c(t_{i+1})$ will then define the $i+1$-th move of the player of the discrete game by enforcing (b) in $T^d(i+1)$. Assume that the i-th choice of the adversary of the discrete game is a leaf $\ell(i)$ with $x_{\ell(i)}$ miners. Then, a similar move is perpetrated in T^c to that same leaf, i.e. edge deletion if $x_{\ell(i)} = 1$, edge elongation if $x_{\ell(i)} = 2$ or fork if $x_{\ell(i)} \geq 3$, so that the property (a) is preserved. Then, while there is a leaf ℓ of T^c satisfying $x_\ell \geq 2$ and $d^c_\ell < d^d_\ell$, that leaf is elongated, leading to the transfer of the excess miner from leaf to leaf. When that property is no longer true, we interrupt the passing of time and this defines t_{i+1} as well as the state of the continuous game $T^c(t_{i+1})$. We then perform the move of the player of the discrete game which enforces (b) at step $i+1$ and note that properties (c) and (d) follow. Properties (a), (b), (c), (d) are thus all satisfied at step $i+1$. It is clear from the description that all moves taking place in T^d up to step i must have taken place in T^c, except for the moves inside partially completed leaves, which came at no cost in the discrete game. In conclusion the cost of the discrete (resp. continuous) game $\mathrm{Cost}^d(i)$ (resp. $\mathrm{Cost}^c(i)$) satisfy $\mathrm{Cost}^d(i) \leq \mathrm{Cost}^c(i)$. ◀

3.3 Layered Tree Traversal

In this section, we first recall the problem of layered tree traversal (LTT) which is a crucial special case of layered graph traversal (LGT) [12]. We then recall the equivalence between fractional strategies and mixed (i.e. randomized) strategies for layered tree traversal [3]. After, we show how a new type of guarantees for layered tree traversal can be obtained from asynchronous collective exploration (ACTE) algorithms.

Layered Tree Traversal. A searcher attempts to go from the source $r \in V$ to the target $r' \in V$ of a weighted tree $T = (V, E)$ with as little effort as possible. The searcher does not initially observe the entire tree T which is instead revealed *layer by layer*. The i-th layer corresponds to the set of nodes of combinatorial depth i. At step i, the i-th layer is revealed along with all the edges going from the $i-1$-th layer to the i-th layer. The searcher then has to choose one node of the i-th layer and must move to this node using previously revealed edges. The length of all edges is assumed to take its values in $\{0, 1\}$. The goal for the searcher is to reach the last layer (which contains r') while traversing the minimum number of length 1 edges. The following quantities are useful to describe the layered tree,

- the *width* $w \in \mathbb{N}$ is equal to the maximum number of nodes of a layer,
- the *depth* $D \in \mathbb{R}$ is equal to the shortest path distance from r to r',
- the *length* $L \in \mathbb{R}$ is equal to the sum of all edge lengths.

It was observed by [12] that the setting of LTT with edge lengths in $\{0, 1\}$ encompasses the general setting of layered tree traversal with arbitrary edge lengths. Given an algorithm $\mathcal{A}$ traversing layered trees with edge lengths in $\{0, 1\}$, we readily obtain an algorithm $\mathcal{A}'$

traversing layered trees with edge lengths in $\mathbb{N}$, by cutting edges in segments of size 0 and 1, inserting intermediary layers when needed. Note that this operation does not change the quantities of interest: the width w, the depth D, and the length L. The argument is easily generalized to allow for edge lengths in $\{0\} \cup [\epsilon, \infty]$ where ϵ is a lower-bound on the smallest non-zero edge length, see [12] for full details.

Fractional strategies and randomized algorithms. A *deterministic algorithm* for layered tree traversal is one that maps a sequence of layers, as well as the position of the searcher in the before-last layer, to the position of the searcher in the last layer. The cost of a deterministic algorithm on some layered tree T equals the total length traversed by a searcher using the algorithm to navigate in that tree, when layers are revealed one after the other. A *randomized algorithm* is a probability distribution over deterministic algorithms, its cost is defined by taking the expectation of the cost over the deterministic algorithms. A *fractional strategy* is one which maps a probability distribution on the before-last layer, to a probability distribution on the last layer. The cost of a fractional strategy is equal to the sum of optimal transport costs between all consecutive probability distributions, until the last layer is revealed. The following reduction was observed by [3].

▶ **Proposition 15.** *For every fractional strategy of the searcher, there is a mixed strategy incurring the same cost in expectation.*

Proof idea. The proof works by induction, the idea is to build a probability distribution over deterministic algorithms that has the same expected cost as the fractional strategy. At each step, the optimal transport cost between two consecutive probability distributions on both layers allows to define a coupling which tells how the position of a searcher should change, given its current position. We refer to [3, Section 4] for more details. ◀

Connection to (ACTE). We now show how an asynchronous collective tree exploration algorithm can be used to obtain a fractional strategy for layered tree traversal (LTT).

▶ **Proposition 16.** *Given an asynchronous collective tree exploration (ACTE) algorithm $\mathcal{B}$, for any $k \in \mathbb{N}$, there exists a mixed strategy of the searcher $\mathcal{A}$ such that for any layered tree T with edge lengths in $\{0, 1\}$, we have,*

$$\mathbb{E}(\texttt{Cost}(\mathcal{A}, T)) \leq \frac{1}{k}\texttt{Moves}(\mathcal{B}, k, T')$$

where T' denotes the tree obtained by concatenating nodes of T connected by length 0 edges.

Proof. We use the asynchronous collective exploration algorithm $\mathcal{B}$ to define a fractional strategy for layered graph traversal. For this purpose we define T' the tree obtained from the weighted tree T by concatenating all nodes that are connected by length 0 edges. We also denote by V'_i the set of all nodes of T' that belong to the i-th layer of T. Note that the sets V'_i no longer form a partition of the nodes, since one node may belong to multiple layers. At each step i, the i-th layer of T is revealed and we shall use our (ACTE) algorithm $\mathcal{B}$ on T' to define a fractional configuration on that layer.

Specifically, we shall consider a run of the algorithm $\mathcal{B}$ on T' and instants $t_0 \leq \cdots \leq t_i \leq \ldots$ such that (1) at time t_i, all robots are located on a node of V'_i (2) before time t_i, no robot located on a node of V'_i was ever granted a move. If these properties are satisfied for some t_i, it is easy to define a time t_{i+1} and a sequence of robot moves such that the property will be satisfied at t_{i+1}. Indeed, simply grant a move to the robots until they each reach a node of V'_{i+1}. Since the exploration algorithm $\mathcal{B}$ finishes in finite time, all robots will eventually reach such node.

Note that the run defined above can be performed online, if the layer $i+1$ is always revealed at time t_i. Thus, this run of $\mathcal{B}$ allows to define a fractional strategy for layered tree traversal, where the probability distribution is given by the distribution of the k robots on V_i', translated in a distribution over V_i which is the actual i-th layer of T. It is clear that the number of robot moves between two consecutive steps is at least k times the optimal transport cost between the distributions. Finally, observe that the exploration is not finished until the last layer, which we assume without loss of generality reduced to the target $\{r'\}$, has been reached by all robots. But the target is then still un-mined because no robot located at r' was granted a move, thus at this instant the total number of moves of the exploration algorithm is less than $\texttt{Moves}(\mathcal{B}, k, T)$. This finishes the proof of the proposition. ◀

4 Applications

In this section, we combine the results obtained for the continuous tree-mining game in Section 2 to the reduction presented in Section 3 in order to obtain new guarantees for collective tree exploration (CTE) and layered tree traversal (LTT).

4.1 Application to Collective Tree Exploration

The first result that we obtain is on asynchronous collective tree exploration (ACTE).

▶ **Theorem 17.** *There exists an asynchronous collective tree exploration algorithm $\mathcal{B}$ satisfying, for any $k \in \mathbb{N}$ and $n \geq D$,*

$$\texttt{Moves}(\mathcal{B}, k, n, D) \leq 2n + \mathcal{O}(k^2 D).$$

Proof. Combine Proposition 14 and Proposition 13 with Theorem 2. ◀

This result then immediately translates to the synchronous setting (CTE), providing the announced competitive ratio.

▶ **Theorem 18.** *There exists a collective tree exploration algorithm $\mathcal{A}$ with runtime,*

$$\texttt{Runtime}(\mathcal{A}, k, n, D) \leq \frac{2n}{k} + \mathcal{O}(kD).$$

The exploration algorithm $\mathcal{A}'$ which uses this algorithm with $k' = \sqrt{k}$ robots while leaving $k - \sqrt{k}$ robots idle at the root thus has a competitive ratio of $\mathcal{O}(\sqrt{k})$.

Proof. Apply Theorem 17 with Proposition 12. For the competitive ratio, observe that

$$\frac{2n}{\sqrt{k}} + \text{cst} \times \sqrt{k} D \leq \text{cst} \times \sqrt{k} \left(\frac{n}{k} + D \right). \quad ◀$$

4.2 Application to Layered Tree Traversal

The main result of this section is the following.

▶ **Theorem 19.** *There exists a collection of layered tree traversal (LTT) randomized algorithms $\{\mathcal{A}_k\}_{k \in \mathbb{N}}$ satisfying for any layered tree T,*

$$\mathbb{E}(\texttt{Cost}(\mathcal{A}_k, T)) \leq \frac{2L}{k} + \mathcal{O}(kD),$$

where L is the sum of all edge lengths in T and D is the distance from source to target.

Proof. Apply Theorem 17 with Proposition 16. ◀

We now discuss the relevance of the above result in light of prior work on layered tree and graph traversal. The most notable difference is that our guarantee does not depend on the width w, contrarily to all previous work on the topic which focused on bounds of the form $\mathcal{O}(c(w)D)$ (competitive analysis). The latest result of this kind is the recent algorithm $\mathcal{A}$ of [3] satisfying, $\mathbb{E}(\texttt{Cost}(\mathcal{A}, G)) \leq \mathcal{O}(w^2 D)$. Instead, our guarantees depend on the sum of all edge lengths L, which cannot be bounded by a function of w and D in full generality. We thus present below two settings illustrating the interest Theorem 19.

Unit edge lengths. Recall that layered tree traversal can be reduced to the case of edge lengths belonging to $\{0, 1\}$. We consider here the simple restriction where the edge lengths are all set equal to 1, and where the environment is thus represented by an unweighted tree structure. This setting seems particularly relevant to robotic applications in which sensors have a given range. The unit lengths assumption entails that $D = N$, where N is the number of layers, and thus that $L \leq wD$, where w is the width of the layered tree. Therefore choosing $k = \lfloor \sqrt{w} \rfloor$, we obtain an algorithm traversing unweighted layered trees of width w with cost $\mathcal{O}(\sqrt{w}D)$. To the best of our knowledge, this is the first guarantee on this problem that improves over the naive $\mathcal{O}(L) = \mathcal{O}(wD)$ upper bound, achieved by a simple depth-first search. More generally, if edges are not of unit lengths, but the ratio between the shortest and the longest edge lengths is bounded by a constant C, we obtain a guarantee in $\mathcal{O}(\sqrt{Cw}D)$.

Average case analysis. Another natural situation which provides a control of the value of L is when the problem instance is sampled at random from probability distribution. One reasonable way to sample a layered graph of width w is as follows. We first pick an arbitrary time horizon N and we consider a set of layers $L_1, \dots, L_N$ that each contain w nodes. One node of the last layer L_N is arbitrarily chosen to be connected to the target r', whereas all nodes of the first layer are connected to the source r. For all $i < N$, we define E_s the set of edges in $L_i \times L_{i+1}$, connecting layer i to layer $i + 1$, by assigning to each node of L_{i+1} one parent in L_i chosen uniformly at random. Then for every edge $e \in E$, we pick an independent sample of a Bernouilli random variable $Z_e \sim \mathcal{B}(\frac{1}{2})$ to decide whether e is of length 0 or 1. The total length of the tree is then the sum of $|E| = wN$ independent Bernoullis, i.e. $L = \sum_{e \in E} Z_e$, and the distance from source to target is the sum of N independent Bernoullis $D = \sum_{e \in P(r,r')} Z_e$ where $P(r, r')$ denotes the path from source to target. In this context, with high probability and for N sufficiently large, L will be of order as $\Theta(wD)$. As above, we then choose $k = \lfloor \sqrt{w} \rfloor$ to obtain a guarantee in $\mathcal{O}(\sqrt{w}D)$. This discussion could be of interest in the perspective of an average case analysis of layered tree and graph traversal.

References

1 Allan Borodin, Nathan Linial, and Michael E Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM (JACM)*, 39(4):745–763, 1992.

2 Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Trans. Robotics*, 27(4):707–717, 2011. doi:10.1109/TRO.2011.2121170.

3 Sébastien Bubeck, Christian Coester, and Yuval Rabani. Shortest paths without a map, but with an entropic regularizer. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1102–1113. IEEE, 2022.

4 William R Burley. Traversing layered graphs using the work function algorithm. *Journal of Algorithms*, 20(3):479–511, 1996.

5 Marek Chrobak and Lawrence L Larmore. The server problem and on-line games. *On-line algorithms*, 7:11–64, 1991.

6 Romain Cosson. Breaking the k/log k barrier in collective tree exploration with tree-mining. In *arXiv preprint arXiv:2309.07011*, 2023. arXiv:2309.07011.

7 Romain Cosson, Laurent Massoulié, and Laurent Viennot. Breadth-first depth-next: Optimal collaborative exploration of trees with low diameter. In *37th International Symposium on Distributed Computing*, 2023.

8 Claude Dellacherie, Servet Martinez, Jaime San Martin, et al. *Inverse M-matrices and ultrametric matrices*, volume 2118. Springer, 2014.

9 Dariusz Dereniowski, Yann Disser, Adrian Kosowski, Dominik Pajak, and Przemyslaw Uznanski. Fast collaborative graph exploration. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 520–532. Springer, 2013. doi:10.1007/978-3-642-39212-2_46.

10 Miroslaw Dynia, Miroslaw Korzeniowski, and Christian Schindelhauer. Power-aware collective tree exploration. In *International Conference on Architecture of Computing Systems*, pages 341–351. Springer, 2006.

11 Miroslaw Dynia, Jakub Lopuszanski, and Christian Schindelhauer. Why robots need maps. In Giuseppe Prencipe and Shmuel Zaks, editors, *Structural Information and Communication Complexity, 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007, Proceedings*, volume 4474 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2007. doi:10.1007/978-3-540-72951-8_5.

12 Amos Fiat, Dean P Foster, Howard Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. *SIAM Journal on Computing*, 28(2):447–462, 1998.

13 Pierre Fraigniaud, Leszek Gasieniec, Dariusz R Kowalski, and Andrzej Pelc. Collective tree exploration. In *LATIN 2004: Theoretical Informatics: 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004. Proceedings 6*, pages 141–151. Springer, 2004.

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

15 Christian Ortolf and Christian Schindelhauer. A recursive approach to multi-robot exploration of trees. In *International Colloquium on Structural Information and Communication Complexity*, pages 343–354. Springer, 2014.

16 Christos H Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

17 H Ramesh. On traversing layered graphs on-line. In *SODA*, pages 412–421, 1993.

18 Bernd Sturmfels, Caroline Uhler, and Piotr Zwiernik. Brownian motion tree models are toric. *arXiv preprint*, 2019. arXiv:1902.09905.

A Proofs of Lemmas 20, 21, 22, 23 on the dynamics of x

▶ **Lemma 20.** *Consider some time interval $I = [t_1, t_2]$, in which some leaf l is continuously extended. In this time interval, all moves by Algorithm* (2) *consist in moving a robot from l to some other leaf of the tree, and no two moves happen simultaneously.*

Proof. We denote by $\boldsymbol{x}$ the configuration of the robots at time t_1 and we assume that it is stable (i.e. for any other configuration $\mathbf{z}$: $\tau(\boldsymbol{x}, \mathbf{z}) < \mathrm{OT}(\boldsymbol{x}, \mathbf{z})$. We denote by $\boldsymbol{x}'$ the first configuration that the player will switch to after $\boldsymbol{x}$, and by $t_3 \in (t_1, t_2)$ the time at which it occurs. We shall show that $\boldsymbol{x}'$ will satisfy $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}_l + \boldsymbol{e}_\ell$ for some leaf $\ell \neq l$. First observe

that before t_3 the tension between any two leaves $\ell, \ell' \neq l$ in $\boldsymbol{x}$ does not change with time (and thus remains always $< d(\ell, \ell')$). Then observe that the tension $\tau(\boldsymbol{x} \to \boldsymbol{x}') = \Phi_t(\boldsymbol{x}') - \Phi_t(\boldsymbol{x})$ evolves continuously with time, as well as $\mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}')$. So at time t_3, when the movement occurs, we have that $\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}') = \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i\in[h]} d(\ell_i, \ell_i')$ for $\ell_1 \to \ell_1', \dots, \ell_h \to \ell_h'$ an optimal transport plan leading from $\boldsymbol{x}$ to $\boldsymbol{x}'$. Using Lemma 4, we get that at time t_3,

$$\sum_{i\in[h]} d(\ell_i, \ell_i') = \tau(\boldsymbol{x} \to \boldsymbol{x}') \leq \tau(\ell_1 \to \ell_1') + \cdots + \tau(\ell_h \to \ell_h'),$$

where by continuity of tensions, since no moves took place before t_3 we have for all $i \in [h]$: $\tau(\ell_i \to \ell_i') \leq d(\ell_i \to \ell_i')$, with strict inequality if $\ell_i \neq l$. This implies that $\forall i \in [h] : \ell_i = l$. Then observe that if $h > 1$, the optimal transport plan has overlaps of positive length, so by the lemma above,

$$\sum_{i\in[h]} d(\ell_i, \ell_i') = \tau(\boldsymbol{x} \to \boldsymbol{x}') < \tau(\ell_1 \to \ell_1') + \cdots + \tau(\ell_h \to \ell_h'),$$

which is impossible using again the continuity of tensions. So it is a necessity that $h = 1$ and that $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}_l + \boldsymbol{e}_{\ell_1'}$. Now to conclude this proof, it suffices to observe that the configuration $\boldsymbol{x}'$ is stable at time t_3 (i.e. for any other configuration $\mathbf{z} : \tau(\boldsymbol{x}', \mathbf{z}) < \mathrm{OT}(\boldsymbol{x}', \mathbf{z})$ where $\mathrm{OT}(\boldsymbol{x}', \mathbf{z})$ denotes the optimal transport distance from $\boldsymbol{x}'$ to $\mathbf{z}$). We can then iterate the argument to show that the property will remain satisfied all the way to time t_2. Assume by contradiction that right after the player makes the move to $\boldsymbol{x}'$ at t_3, there is another configuration $\mathbf{z} \neq \boldsymbol{x}'$ satisfying $\tau(\boldsymbol{x}', \mathbf{z}) \geq \mathrm{OT}(\boldsymbol{x}', \mathbf{z})$, then the following identities held before the move at time t_3, $\tau(\boldsymbol{x}, \mathbf{z}) = \tau(\boldsymbol{x}, \boldsymbol{x}') + \tau(\boldsymbol{x}', \mathbf{z}) \geq \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') + \mathrm{OT}(\boldsymbol{x}', \mathbf{z}) \geq \mathrm{OT}(\boldsymbol{x}, \mathbf{z})$. So $\mathbf{z}$ would have been another allowed candidate at time t_3. Thus, $\mathbf{z} = \boldsymbol{x} - \boldsymbol{e}_l + \boldsymbol{e}_{\ell'}$ for some other leaf ℓ'. This implies that $\mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') + \mathrm{OT}(\boldsymbol{x}', \mathbf{z}) > \mathrm{OT}(\boldsymbol{x}, \mathbf{z})$ which in turns implies that at time t_3, before the move of the player we had $\tau(\boldsymbol{x}, \mathbf{z}) > \mathrm{OT}(\boldsymbol{x}, \mathbf{z})$, which is a contradiction. ◀

▶ **Lemma 21.** *When some leaf l is deleted, denoting by $\boldsymbol{x}$ the configuration before the deletion and by $\boldsymbol{x}'$ the configuration right after the deletion, we have that $\forall \ell \neq l : y_\ell \geq x_\ell$. In other words, all moves by Algorithm* (2) *are from the deleted leaf to other leaves of the tree.*

Proof. We denote by $\boldsymbol{x}$ the configuration before the removal of l and by $\boldsymbol{x}'$ the configuration after the removal of l and we recall that the transport distance $\mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i\in[h]} d(\ell_i, \ell_i')$ where $\ell_1 \to \ell_1', \dots, \ell_h \to \ell_h'$ is an optimal transport plan leading from $\boldsymbol{x}$ to $\boldsymbol{x}'$. We also recall that Lemma 4 gives,

$$\tau(\boldsymbol{x} \to \boldsymbol{x}') \leq \tau_{\boldsymbol{x}}(\ell_1 \to \ell_1') + \cdots + \tau_{\boldsymbol{x}}(\ell_h \to \ell_h').$$

We assume by contradiction that there is a leaf $\ell \neq l$ satisfying $y_\ell < x_\ell$. Without loss of generality, this implies that we can choose a leaf $\ell_i \neq l$ of the optimal transport plan, for $i \in [h]$, such that $y_{\ell_i'} > x_{\ell_i'}$. Consequently the configuration $\boldsymbol{x}'' = \boldsymbol{x}' - \boldsymbol{e}_{\ell_i'} + \boldsymbol{e}_{\ell_i}$ where one move from ℓ_i to ℓ_i' never took place is valid, even after the deletion of leaf l because obviously $\ell_i' \neq l$. Also remember that by the proof of Lemma 4,

$$\tau(\boldsymbol{x} \to \boldsymbol{x}') \leq \tau(\boldsymbol{x} \to \boldsymbol{x}'') + \tau(\ell \to \ell')$$

and recall,

$$\mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}'') + d(\ell, \ell').$$

Since $\boldsymbol{x}'$ was performed instead of $\boldsymbol{x}''$, it must have been that $\tau(\boldsymbol{x} \to \boldsymbol{x}') - \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}') \geq \tau(\boldsymbol{x} \to \boldsymbol{x}'') - \mathrm{OT}(\boldsymbol{x}, \boldsymbol{x}'')$ and thus that $\tau_{\boldsymbol{x}}(\ell \to \ell') \geq d(\ell, \ell')$. This is not possible, because it implies that a movement from ℓ to ℓ' would have taken place before the removal of l, at time t_1^-. ◀

▶ **Lemma 22.** *Consider the fork of a leaf l in m children, and denote by $\boldsymbol{x}$ the configuration right before the fork. We call "configuration induced by the fork" and we denote by $\boldsymbol{x}'$ a configuration in which the x_l robots formerly located at l are partitioned evenly on the m children of l, i.e. where the newly created children take values in $\{\lceil \frac{x_l}{m} \rceil, \lfloor \frac{x_l}{m} \rfloor\}$ and where $x'_l = x_l$. There exists some $\delta \in (0, 1)$, such that for any fork length of δ, the configuration $\boldsymbol{x}'$ is stable after the fork.*

Proof. We recall that a fork of leaf l in m children leads to the creation of m new leaves at distance δ from l. By convexity of $\Phi(\cdot)$, the configuration $\boldsymbol{x}'$ defined above are such that for any ℓ, ℓ' newly created children, $\tau_{\boldsymbol{x}'}(\ell \to \ell') \leq 0$ and thus, $\tau_{\boldsymbol{x}'}(\ell \to \ell') - d(\ell, \ell') < 0$ after the fork. We consider $T(u)$ the tree corresponding to a fork length of $u \in [0, 1)$. We note that for any two leaves ℓ, ℓ', which are not both children of l, the value of $\tau_{\boldsymbol{x}'}(\ell \to \ell') - d(\ell, \ell')$ evolves continuously and is known to be strictly below 0 for $u = 0$, by stability of the configuration $\boldsymbol{x}$ before the fork. Therefore there is a value of $\delta > 0$, such that the benefit of moving robots between leaves remains below below 0, and by Lemma 4, this implies that the configuration $\boldsymbol{x}'$ defined above is stable after the fork. ◀

The lemmas above, also lead to the following result.

▶ **Lemma 23.** *Over the course of the evolving tree game, the discrete configuration $\boldsymbol{x}$ always satisfies $\forall \ell \in \mathcal{L} : x_\ell \geq 1$.*

Proof. We assume by contradiction that the property is not satisfied, and we consider the first instant t when a leaf was depopulated for the first time (take the supremum over the instants for which the property hold). By Lemma 20, t cannot occur in the middle of a leaf elongation because a leaf with a single robot cannot be elongated. It must thus correspond to the instant of a discrete step. But by Lemma 21 and 22 no discrete steps can lead to a lonely leaf. This finishes the contradiction. ◀