A general lower bound for collaborative tree exploration

Yann Disser[1⋆], Frank Mousset[2⋆⋆], Andreas Noever[2⋆⋆⋆], Nemanja Škorić[2], and Angelika Steger[2]

[1] Institute of Mathematics, Graduate School CE, TU Darmstadt, Germany
disser@mathematik.tu-darmstadt.de
[2] Department of Computer Science, ETH Zurich, Switzerland
{moussetf|anoever|nskoric|steger}@inf.ethz.ch.

Abstract. We consider collaborative graph exploration with a set of k agents. All agents start at a common vertex of an initially unknown graph with n vertices and need to collectively visit all other vertices. We assume agents are deterministic, moves are simultaneous, and we allow agents to communicate globally. For this setting, we give the first non-trivial lower bounds that bridge the gap between small ($k \leq \sqrt{n}$) and large ($k \geq n$) teams of agents. Remarkably, our bounds tightly connect to existing results in both domains.
First, we significantly extend a lower bound of $\Omega(\log k / \log\log k)$ by Dynia et al. on the competitive ratio of a collaborative tree exploration strategy to the range $k \leq n \log^c n$ for any $c \in \mathbb{N}$. Second, we provide a tight lower bound on the number of agents needed for any competitive exploration algorithm. In particular, we show that any collaborative tree exploration algorithm with $k = Dn^{1+o(1)}$ agents has a competitive ratio of $\omega(1)$, while Dereniowski et al. gave an algorithm with $k = Dn^{1+\varepsilon}$ agents and competitive ratio $\mathcal{O}(1)$, for any $\varepsilon > 0$ and with D denoting the diameter of the graph. Lastly, we show that, for any exploration algorithm using $k = n$ agents, there exist trees of arbitrarily large height D that require $\Omega(D^2)$ rounds, and we provide a simple algorithm that matches this bound for all trees.

1 Introduction

Graph exploration captures the problem of navigating an unknown terrain with a single or multiple autonomous robots. In the abstract setting, we take the perspective of an agent that is located at some vertex of an initially unknown graph, can locally distinguish edges at its current location, and can choose an edge to traverse in its next move. Various scenarios for graph exploration have been studied in the past, for different graph classes and different capabilities of the agent(s). A fundamental goal of exploration is to systematically visit all vertices/edges of the underlying graph. For settings where exploration is possible, we typically ask for efficient exploration algorithms, e.g., in terms of the number of edge traversals.

In this paper, we consider *collaborative* exploration, where a set of k agents are initially located at some vertex of an unknown *undirected* graph. We assume agents to move deterministically, allow them to freely communicate at all times, and to have unlimited computational power and memory at their disposal. In every round each agent may traverse any edge incident to its current location, where the edges incident to a vertex are revealed when that vertex is visited for the first time. The goal is to visit all vertices while minimizing the number of rounds. More precisely, we are interested in the competitive ratio of an exploration strategy, i.e., the worst case ratio between the total number of rounds it needs and the minimum total number of rounds needed to visit all vertices of the same graph, assuming it is known beforehand. We prove new lower bounds for the best-possible competitive ratio of any collaborative exploration algorithm. Our bounds hold even for the much simpler setting of *tree* exploration. Note that since our results concern trees, it makes no difference whether nodes can be distinguished, and whether the agents need to visit all edges or not.

⋆ Supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School CE at TU Darmstadt.
⋆⋆ Supported by grant no. 6910960 of the Fonds National de la Recherche, Luxembourg.
⋆⋆⋆ Supported by grant no. 200021 143338 of the Swiss National Science Foundation.

Let $\mathcal{T}_{n,D}$ denote set of all rooted trees with n vertices and height D. Each such tree corresponds to an instance of the tree exploration problem in which all k agents start at the root of the tree. Clearly, any offline exploration algorithm needs $\Omega(n/k + D)$ rounds to explore a tree in $\mathcal{T}_{n,D}$ using k agents. This is shown to be tight by the following offline exploration algorithm that explores the tree in $\Theta(n/k + D)$ rounds: start with the tree T, double its edges, find an Eulerian tour C (of length $2n-2$), distribute the agents evenly on C (this takes at most D rounds), and explore T by letting each agent walk along C for $\mathcal{O}(n/k)$ rounds.

In the online setting, we can explore a tree in $\mathcal{T}_{n,D}$ with a single agent using a depth-first traversal in time $\mathcal{O}(n)$ and thus we trivially have a competitive ratio of $\mathcal{O}(1)$ when k is constant. On the other hand, with $k \geq \Delta^D$ agents, where Δ is the maximum degree of the tree, we can simply perform a breadth-first traversal, which takes $\mathcal{O}(D)$ steps and thus also has competitive ratio $\mathcal{O}(1)$. Observe that in the first case n/k dominates the lower bound on the offline optimum, while in the second case D is dominating. We are interested in the best-possible competitive ratios between these two extreme cases.

Surprisingly, Dereniowski et al. [12] showed that already a polynomial number $k = Dn^{1+\varepsilon}$ of agents allows for a BFS-like algorithm that achieves a constant competitive ratio. For smaller teams of agents, Fraigniaud et al. [18, 20] gave a collaborative algorithm with competitive ratio $\mathcal{O}(k/\log k)$. This is only slightly better than the trivial upper bound of $\mathcal{O}(k)$ that we get by performing a depth first traversal with a single agent. Ortolf and Schindelhauer [23] improved this competitive ratio to $k^{o(1)}$ for $k = 2^{\omega(\sqrt{\log D \log\log D})}$ and $n = 2^{\mathcal{O}(2^{\sqrt{\log D}})}$. The only non-trivial lower bound for collaborative tree exploration was given by Dynia et al. [16]. They showed that any deterministic exploration algorithm for $k < \sqrt{n}$ agents has competitive ratio $\Omega(\log k/\log\log k)$.

Our Results

We give the first non-trivial lower bounds on the competitive ratio for collaborative tree exploration in the domain $k \geq \sqrt{n}$ (cf. Figure 1). More precisely, we show that for every constant $c \in \mathbb{N}$, any given deterministic exploration strategy with $k \leq n \log^c n$ agents has competitive ratio $\Omega(\log k/\log\log k)$ on the set of all trees on n vertices. Note that this extends the range of the bound by Dynia et al. [16] for $k < \sqrt{n}$ significantly.

Secondly, we show that for every constant $\varepsilon > 0$, there is a constant $D = D(\varepsilon)$ such that for any exploration algorithm with $k \leq Dn^{1+\varepsilon}$ agents, there exists a tree in $\mathcal{T}_{n,D}$ on which the algorithm needs at least $D/(5\varepsilon)$ rounds. This (almost) tightly matches the algorithm of Dereniowski et al. [12], which can explore any tree in at most $(1 + o(1))D/\varepsilon$ rounds using $k = Dn^{1+\varepsilon}$ agents. Our result implies that any exploration algorithm with $k = Dn^{1+o(1)}$ agents has competitive ratio $\omega(1)$. More precisely, we get that for any function $0 \leq f(n) \leq o(1)$, there is a function $D = D(n)$ such that every exploration algorithm with $k = Dn^{1+f(n)}$ agents has competitive ratio $\omega(1)$ on the trees in $\mathcal{T}_{n,D}$. In contrast, the algorithm of Dereniowski et al. shows that $k = Dn^{1+\varepsilon}$ agents are sufficient to get a competitive ratio $\mathcal{O}(1)$ on such trees.

Finally, for every exploration algorithm with $k = n$, we construct a tree of height $D = \omega(1)$ where the algorithm needs $\mathcal{O}(D^2)$ rounds. We give a simple algorithm that achieves this bound in general.

Further Related Work

Many variants of graph exploration with a *single agent* have been studied in the past. Any (strongly) connected graph with *distinguishable* vertices can easily be explored in polynomial time by systematically building a map of the graph. Regarding the exploration of *undirected* graphs with *indistinguishable* vertices, Aleliunas et al. [2] showed that a random walk explores any graph in $\mathcal{O}(n^3\Delta^2 \log n)$ steps, with high probability. In order to turn this into a terminating exploration algorithm the agent needs $\Omega(\log n)$ bits of memory. Fraigniaud et al. [19] showed that every deterministic algorithm needs $\Omega(\log n)$ bits of memory, and Reingold [25] gave a matching upper bound. Disser et al. [14] showed that alternatively $\Theta(\log\log n)$ pebbles and bits of memory are necessary and sufficient for exploration, where a pebble is a device that can be dropped to make a vertex distinguishable and that can be picked up

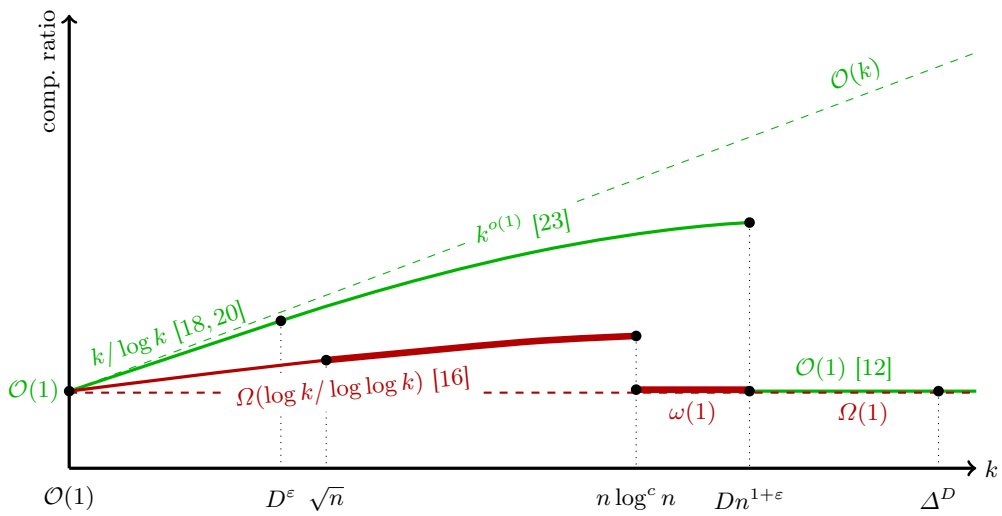

Fig. 1. State of the art in collaborative tree exploration. The top curve shows upper and the bottom curve lower bounds. Thick lines show our results.

and reused later. Diks et al. [13] showed that trees can be explored with $\mathcal{O}(\log \Delta)$ memory, and that $\Omega(\log n)$ memory is required if the agent needs to eventually terminate at the start vertex. Ambühl [4] gave a matching upper bound for the latter result.

For the case of *directed* graphs with *distinguishable* vertices, Albers and Henzinger [1] gave an exploration algorithm with subexponential running time $d^{\mathcal{O}(\log d)}m$ that learns a map of the graph. Here m denotes the number of edges and d is the deficiency of the graph, i.e., the number of edges missing to make the graph Eulerian. This results narrows the gap between a quadratic lower bound and an exponential upper bound introduced by Deng and Papadimitriou [11].

An even more challenging setting (for the agent) is the exploration of *directed*, strongly connected graphs with *indistinguishable* vertices. In general the agent needs exponential time to explore a graph in this setting. On the other hand, Bender and Slonim [7] showed that two agents can explore any directed graph in polynomial time, using a randomized strategy. Bender et al. [6] showed that to accomplish this with a single agent we need $\Theta(\log\log n)$ pebbles, i.e., "a friend is worth $\mathcal{O}(\log\log n)$ pebbles". Remarkably, Bender et al. [6] also showed that if the number of vertices is known beforehand, a deterministic agent with a single pebble can explore any directed graph in polynomial time $\mathcal{O}(n^8\Delta^2)$.

The lower bounds for collaborative tree exploration discussed above carry over to the *collaborative exploration* of general undirected graphs with distinguishable vertices. Also, the algorithm of Dereniowski et al. [12] for $k = Dn^{1+\varepsilon}$ works on general graphs. Additionally, Ortolf and Schindelhauer [22] gave a lower bound on the best-possible competitive ratio for randomized algorithms of $\Omega(\sqrt{\log k}/\log\log k)$ for $k = \sqrt{n}$. Collaborative exploration by multiple random walks without communication has been considered by Alon et al. [3], Elsässer and Sauerwald [17], and Ortolf and Schindelhauer [24].

Graph exploration has been studied in many other settings. Examples include tethered exploration or exploration with limited fuel [5, 15], exploration of mazes [8, 21], and exploration of polygonal environments [9, 10].

2 Results

Our first result extends the lower bound for $k < \sqrt{n}$ agents of Dynia et al. [16] to the much larger range $k \leq n\log^{\mathcal{O}(1)} n$. We prove the following theorem:

Theorem 1. *Let c be any positive integer constant. Then for every n and every $1 \leq k \leq n \log^c n$ there is some $D = D(n, k, c)$ such that the following holds: for any given deterministic exploration strategy with k agents, there exists a tree T on n vertices and with height D on which the strategy needs*

$$\Omega\Big(\frac{\log k}{\log \log k} \cdot (n/k + D) \Big)$$

rounds.

As mentioned above, there is an offline algorithm that explores any graph with n vertices and height D in time $\Theta(n/k + D)$. From this, we obtain the following corollary to Theorem 1:

Corollary 1. *Let c be any positive integer constant. Then any deterministic exploration strategy using $k \leq n \log^c n$ agents has a competitive ratio of*

$$\Omega\Big(\frac{\log k}{\log \log k} \Big).$$

Our second main result shows that the algorithm of Dereniowski et al. [12] that explores a graph with $k = Dn^{1+\varepsilon}$ agents in time $(1+o(1))D/\varepsilon$ is almost optimal: using $k \leq Dn^{1+\varepsilon}$ agents it is generally impossible to explore the graph in fewer than $D/(5\varepsilon)$ rounds.

Theorem 2. *Given any constant $\varepsilon > 0$ there is an integer $D = D(\varepsilon)$ such that for sufficiently large n and for every deterministic exploration strategy using $k \leq D \cdot n^{1+\varepsilon}$ agents, there exists a tree on n vertices and with height D on which the strategy needs at least $D/(5\varepsilon)$ rounds.*

In the range where $k \geq n$, the offline optimum is determined by the height D of the tree. Therefore, the result of Dereniowski et al. mentioned above implies that the competitive ratio is constant when $k = D \cdot n^{1+\Omega(1)}$. Theorem 2 shows in particular that this is tight in the following sense:

Corollary 2. *For any function $0 \leq f(n) \leq o(1)$, there is a function $D = D(n)$ such that the competitive ratio of any deterministic exploration strategy using $k = D \cdot n^{1+f(n)}$ agents is $\omega(1)$ on the the set $\mathcal{T}_{n,D}$ of all rooted trees with n vertices and height D.*

However, note that here we have no control over the height of the worst-case example: for instance, it could be that there are ranges for D where the algorithm of Dereniowski et al. may be improved.

Finally, it is possible for $k = n$ agents to explore any tree on n vertices and of height D in D^2 rounds using a breadth-first exploration strategy. More precisely, we can split the D^2 rounds in D phases of length D, and in each phase $1 \leq i \leq D$ do the following. Let A_i be the set of unvisited leaves of the tree that is known to the agents at the start of phase i. Then we send one agent to each vertex in A_i along a shortest path. This is clearly doable in D rounds, and constitutes a single phase. After phase i, the agents have explored all vertices at distance at most i from the root. Therefore, after D such phases, the tree is completely explored. We show that the running time of D^2 is optimal up to a constant factor:

Theorem 3. *For every n and every deterministic exploration strategy using $k = n$ agents, there exists a tree T on n vertices and with height $D = \omega(1)$ such that the strategy needs at least $D^2/3$ rounds to explore T.*

In all the results above, we have considered the worst-case performance of an exploration strategy on any tree. However, by looking at the proofs of Theorems 1 and 2, one can see that the heights of our lower bound constructions are typically quite small. We believe it is also natural to ask about the competitive ratio on the set of trees of height at least D, for a given D. We show that at least for subpolynomial heights, the competitive ratio with $k = \Theta(n)$ agents is unbounded:

Theorem 4. *For any function $D \leq n^{o(1)}$ and any exploration strategy using $k = \Theta(n)$ agents, the competitive ratio on the set of all trees of size n and height at least D is $\omega(1)$.*

We stress that Theorem 4 differs from the other results in that it gives a measure of control over the height of the adversarial example, while the other results merely state that there *exists* some height on which the algorithm must perform poorly.

3 Tree exploration games

In order to prove a lower bound on the competitive ratio, we consider a tree exploration game defined as follows. By a *tree exploration game with k agents* we mean a game with two players, the *explorer* (the online algorithm) and the *revealer* (the adversary), played according to the following rules. The game proceeds in rounds which we index by the variable t ('time'), the first round being $t = 0$. The state of the game at time t is described by a triple (T_t, A_t, ϕ_t), where T_t is a rooted tree (the tree revealed at the beginning of round t), A_t is a subset of the vertices of T_t (the subset of *visited* vertices by round t), and $\phi_t \colon \{1, \ldots, k\} \to A_t$ is an assignment of the agents to the vertices (where $\phi_t(i)$ is the location of the i-th agent at time t). In round $t = 0$ the revealer decides on the initial tree T_0. The state at time 0 is then given by (T_0, A_0, ϕ_0) where $A_0 = \{\mathrm{root}(T_0)\}$ and $\phi_0(x) = \mathrm{root}(T_0)$ for all $1 \leq x \leq k$ – that is to say, all agents are initially at the root of T_0. In every round $t > 0$, each player can make a move. First, the explorer creates a new assignment ϕ_t by moving each agent i to a neighbor of $\phi_{t-1}(i)$ in T_{t-1} or by keeping the location of the agent same, i.e., $\phi_t(i) = \phi_{t-1}(i)$. Then the revealer decides on the new tree T_t, where T_t must be obtained from T_{t-1} by attaching (possibly empty) trees at some vertices $v \in V(T_{t-1}) \setminus A_{t-1}$, where $V(T_{t-1})$ is the set of vertices of T_{t-1}. We then let $A_t = A_{t-1} \cup N_t$ where $N_t = \{\phi_t(i) : 1 \leq i \leq k\}$ is the set of the new agent locations. The game ends in round t^* if all vertices of T_{t^*} are visited at the beginning of round t^*, i.e., if $A_{t^*} = V(T_{t^*})$.

This type of game naturally lends itself to proving lower bounds for the time in which k agents can explore an unknown tree. Specifically, consider any deterministic strategy for exploring an unknown tree T with k agents. Such a strategy can be interpreted as a strategy for the explorer in the tree exploration game with k agents. If the revealer can play so that the game lasts for at least t^* rounds, then this means that the proposed exploration strategy needs t^* rounds to explore the tree T_{t^*}. We will use this observation to prove lower bounds for the online graph exploration in the following section.

As a side remark, here it is crucial that the strategy is deterministic: if the strategy were allowed to make random choices, then the tree T_{t^*} would turn out to be a random variable that might be highly correlated with the random choices made by the explorer, and it could not serve as an instance on which the strategy performs badly.

4 Lower bound construction

We now give our lower bound construction that establishes the following technical lemma.

Lemma 1. *Let n, L, m be positive integers such that $n \geq L \cdot 16^m$. Then for any deterministic exploration strategy using*

$$k \leq \frac{n^{1+1/m}}{6L(m+1)^2(2L)^{1/m}}$$

agents, there exists a tree T on n vertices and of height Lm such that the strategy needs at least $L\binom{m}{2}$ rounds to explore T.

The parameter L is mostly there to force a large diameter. For a first understanding it does not hurt to imagine that $L = 1$ and to think of m as being a function tending to infinity very slowly as n grows. Then the lemma shows that $n^{1+o(1)}$ vertices need $\omega(1)$ rounds to explore the tree.

Proof. Assume that integers n, L and m as above are given. Let k be any integer such that $1 \leq k \leq n^{1+1/m}/(6L(m+1)^2(2L)^{1/m})$. To prove the lemma, we will describe a strategy for the revealer in the tree exploration game with k agents such that

- the game does not end before round $t^* := L \cdot \binom{m}{2}$, and
- the tree T_{t^*} has height Lm and at most n vertices,

where the notation is as in Section 3. Note that this is enough to prove the lemma.

The main difficulty is that there are several trade-offs involved. On the one hand, the game has to keep going for t^* rounds, that is, it must not happen that the agents explore the whole tree at any time before t^*. This requires us to grow the tree at several critical times, when the agents may come close to exploring everything. On the other hand, we do not want to grow the tree too often, or too much, because the final tree must consist of at most n vertices. Lastly, there is the (less severe) constraint that we want the constructed tree to have a certain height, which we must keep in mind.

Before explaining the strategy, we fix some notation. Let

$$\alpha := (2L/n)^{1/m} \quad \text{and} \quad t_i := L \cdot \binom{i+1}{2}$$

for $0 \leq i < m$. For each $t \geq 0$ we can consider the equivalence relation $\sim_t$ on $V(T_t)$ where $u \sim_t v$ if there exists a path between u and v in T_t that avoids the root of T_t (i.e., if they have a common ancestor that is not the root). Since $T_0 \subseteq T_1 \subseteq T_2 \subseteq \dots$ are trees with the same root, we will just write $u \sim v$ instead of $u \sim_t v$ without causing confusion. Then we define $a_t(v) := |\{x \mid \phi_t(x) \sim v\}|$. In other words, $a_t(v)$ counts the total number of agents that could reach vertex v without passing through the root (under the assignment ϕ_t). We think of those agents as being 'near' the vertex v.

The times $t_1, t_2, t_3, \dots$ are our 'critical times' at which the tree grows. The general idea is very natural: at every critical time t_i, we grow the tree in those places where there are the fewest agents nearby. When doing this, we add sufficiently many vertices so that the agents that are currently nearby cannot explore the newly added subtrees in before the next critical time t_{i+1}. Similarly, everything is set up so that the agents that are not nearby are unable to reach the location before the time t_{i+1}. Thus, the game keeps going until round t_{i+1}. The parameter α enforces a sort of 'iterative thinning' of the tree, which allows us to be economical with the vertices. A precise description of the revealing strategy is given in Algorithm 1.

Algorithm 1: The strategy for the revealer.

begin
  let T_0 be a 'star' consisting of $\lceil n/(2L) \rceil$ paths of length L from the root;
  foreach *round* $t = 1, 2, 3, \dots$ **do**
    let the explorer choose ϕ_t;
    if $t = t_i$ *for some* $1 \leq i < m$ **then**
      let K_i be a maximal set of vertices in $V(T_{t_i-1}) \setminus A_{t_i-1}$ s.t.

        (i) every vertex in K_i has distance $L \cdot i$ to the root in T_{t_i-1}
        (ii) there are no two distinct vertices $u, v \in K_i$ with $u \sim v$.

      let $S_i \subseteq K_i$ be the $\lceil \alpha |K_i| \rceil$ vertices $v \in K_i$ with least $a_{t_i}(v)$;
      define T_{t_i} by attaching at each $v \in S_i$ a path of length $L-1$ with a star with $L \cdot (i+1) \cdot a_{t_i}(v)$ leaves at the end;
    else
      let $T_t = T_{t-1}$;
    end
  end
end

The set S_i is the set of vertices where we grow the tree at time t_i. For a better intuition, we refer the reader to Figure 2, which shows what the tree constructed by this strategy might look like. We establish three claims which are used to show that the algorithm indeed runs for at least $t^* = t_{m-1}$ rounds and that the tree constructed in this way has the right properties.

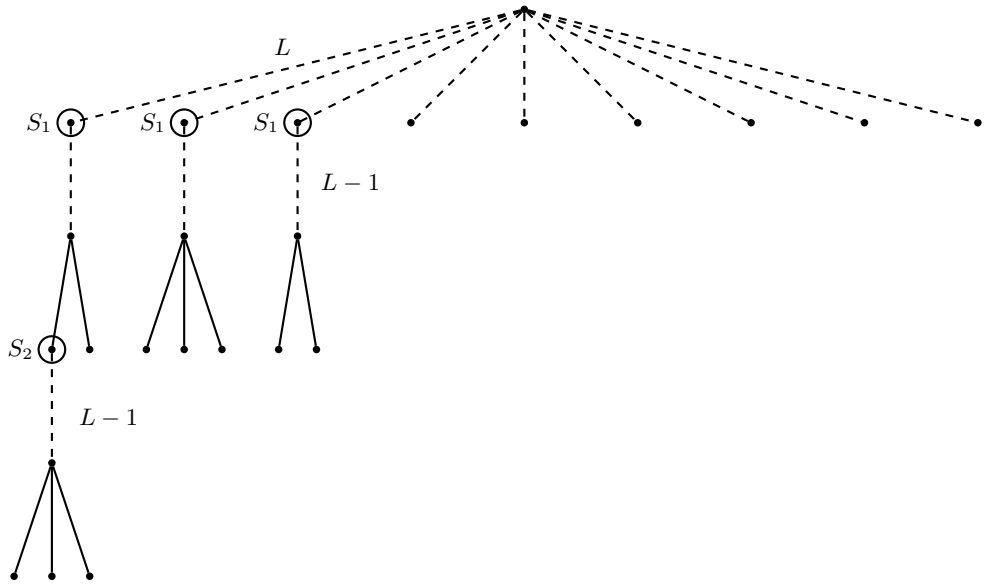

Fig. 2. A sketch of the tree generated by the revealing strategy, for artificial values $\alpha = 1/3$ and $\lceil n/(2L) \rceil = 9$ (degrees in the actual construction are much larger). The actual shape depends on the distribution of the agents at times t_1, t_2. Dashed lines represent paths of the specified length.

Claim. For every $0 \le i < m$ the following holds. The height of T_{t_i} is at most $L \cdot (i+1)$. Moreover, if $S_1, \ldots, S_i$ are all non-empty, then the height of T_{t_i} is exactly $L \cdot (i+1)$.

Proof. The tree T_{t_i} differs from $T_{t_{i-1}}$ if and only if S_i is non-empty, and in this case it is obtained by attaching trees of height L at some vertices with distance $L \cdot i$ to the root in $T_{t_{i-1}}$. Since $T_{t_0} = T_0$ has height L, this implies the claim by induction. □

Claim. For all $1 \le i < m$ and every $v \in S_i$, there exists at least one descendant of v at depth $L \cdot (i+1)$ in $T_{t_{i+1}-1}$ that does not belong to $A_{t_{i+1}-1}$. In particular, for all $1 \le i < m$ we have $|K_{i+1}| = |S_i|$.

Proof. Each vertex at depth $L(i+1)$ is a descendant of some vertex $v \in S_i$. Moreover, we have $u \nsim v$ for any two distinct $u, v \in S_i$. Thus, the second claim follows directly from the first.

For the first claim, consider any $1 \le i < m$ and $v \in S_i$. Note that

(1) at time t_i we create $L \cdot (i+1) \cdot a_{t_i}(v)$ descendants of v at depth $L \cdot (i+1)$;
(2) $t_{i+1} - t_i = L \cdot (i+1)$.

Because of this, no agent passing through the root can visit any descendant of v at depth $L \cdot (i+1)$ before round t_{i+1}. On the other hand, the $a_{t_i}(v)$ agents that could visit a descendant at this depth without passing through the root cannot visit *all* descendants before round t_{i+1}. Thus at least one descendant at depth $L \cdot (i+1)$ must be unvisited at the end of round $t_{i+1} - 1$. □

Claim. For every $1 \le i < m$ we have the bounds

$$|S_i| \ge \frac{\alpha^i n}{2L} \ge \frac{1}{\alpha} \quad \text{and} \quad |S_i| \le \frac{(2\alpha)^i n}{2L}.$$

Proof. By definition we have $\alpha = (2L/n)^{1/m} < 1$ and thus $\alpha^m = 2L/n$, which gives us

$$\frac{\alpha^i n}{2L} \geq \frac{\alpha^{m-1} n}{2L} = 1/\alpha$$

for all $1 \leq i < m$.

For the lower bound, note that since A_0 contains only the root, we have $|K_1| = \lceil n/(2L) \rceil$. By the definition of S_i, we have $|S_i| \geq \alpha |K_i|$ for all $1 \leq i < m$. Moreover, if $2 \leq i < m$ then by Claim 4 we have $|K_i| = |S_{i-1}|$. The lower bound then follows by induction.

For the upper bound, note that $K_1 \leq n/(2L) + 1 \leq n/L$, where the last inequality uses $n \geq 2L$. Moreover, using $|K_i| \geq 1/\alpha$ we have $|S_i| \leq \alpha |K_i| + 1 \leq 2\alpha |K_i|$ for all $1 \leq i < m$. Finally, if $2 \leq i < m$ then $|K_i| = |S_{i-1}|$ by Claim 4, and the upper bound follows by induction. □

Since $|S_i| > 0$ implies in particular that $A_{t_i - 1} \neq V(T_{t_i - 1})$, we conclude from Claim 4 that the game does not stop before reaching round $t_{m-1} = L \cdot \binom{m}{2} = t^*$. Moreover, from Claim 4 and Claim 4 we see that $T_{t_{m-1}}$ is a tree with height $L \cdot m$. To complete the proof we need to show that $|V(T_{t_{m-1}})| \leq n$. We have

$$\begin{aligned} |V(T_{t_{m-1}})| &\leq \lceil n/(2L) \rceil \cdot L + 1 + \sum_{i=1}^{m-1} \sum_{v \in S_i} (L - 1 + L \cdot (i+1) \cdot a_{t_i}(v)) \\ &\leq n/2 + L + 1 + \sum_{i=1}^{m-1} L(i+1) \sum_{v \in S_i} a_{t_i}(v) + \sum_{i=1}^{m-1} |S_i|(L-1). \end{aligned} \tag{1}$$

To bound the double sum note that $|K_i| \geq |S_i| \geq 1/\alpha$ (Claim 4) implies that $\lceil \alpha |K_i| \rceil \leq 2\alpha |K_i|$. Note also that the sum $\sum_{v \in K_i} a_{t_i}(v)$ in (1) is at most k, as no two vertices u, v from K_i are in the same subtree, i.e., $u \not\sim v$. Since S_i contains the $\lceil \alpha |K_i| \rceil \leq 2\alpha |K_i|$ vertices of K_i with least $a_{t_i}(v)$, we thus have

$$\sum_{v \in S_i} a_{t_i}(v) \leq 2\alpha \sum_{v \in K_i} a_{t_i}(v) \leq 2\alpha k,$$

and therefore

$$\sum_{i=1}^{m-1} L(i+1) \sum_{v \in S_i} a_{t_i}(v) \leq L(m+1)^2 \alpha k. \tag{2}$$

To bound the simple sum in (1), we use the upper bound from Claim 4 and obtain

$$\sum_{i=1}^{m-1} |S_i|(L-1) \leq (L-1) \sum_{i=1}^{\infty} \frac{(2\alpha)^i n}{2L} = \frac{L-1}{2L} \cdot 2\alpha n \sum_{i=0}^{\infty} (2\alpha)^i \leq \frac{2\alpha n}{2 - 4\alpha}. \tag{3}$$

Combining (1) with (2) and (3), we get

$$|V(T_{t_{m-1}})| \leq n/2 + L + 1 + L(m+1)^2 \alpha k + \frac{2\alpha n}{2 - 4\alpha}. \tag{4}$$

Since $n \geq L \cdot 16^m \geq 12L$ we have $L + 1 \leq 2L \leq n/6$. By the definition $\alpha = (2L/n)^{1/m}$ and the assumption $k \leq n^{1+1/m}/(6L(m+1)^2(2L)^{1/m})$ we have

$$L(m+1)^2 \alpha k = L(m+1)^2 (2L/n)^{1/m} k \leq n/6.$$

Finally, $n \geq L \cdot 16^m$ implies that $\alpha \leq 1/8$ and so the last term in (4) is also at most $n/6$. Hence $|V(T_{t_{m-1}})| \leq n/2 + 3n/6 = n$. □

5 Consequences for competitiveness

We now use Lemma 1 to derive consequences for best-possible competitive ratios of collaborative tree exploration algorithms. In the proofs below, log is always to the natural base e.

Theorem 1. *Let c be any positive integer constant. Then for every n and every $1 \leq k \leq n \log^c n$ there is some $D = D(n,k,c)$ such that the following holds: for any given deterministic exploration strategy with k agents, there exists a tree T on n vertices and with height D on which the strategy needs*

$$\Omega\Big(\frac{\log k}{\log\log k} \cdot (n/k + D) \Big)$$

rounds.

Proof. By the result of Dynia et al. [16] it suffices to consider the case where $k \geq \sqrt{n}$. Let $c > 0$ be a constant and assume $k \leq n \log^c n$. We apply Lemma 1 with $m = \lceil \frac{\log n}{(8+c)\log\log n} \rceil$ and $L = \lceil n/(mk) \rceil$. Using $k \geq \sqrt{n}$, we have $L = \mathcal{O}(\sqrt{n})$ and $m = o(\log n)$ and thus $n \geq L \cdot 16^m$ holds for sufficiently large n. The lemma states that if

$$k \leq \frac{n^{1+1/m}}{6L(m+1)^2(2L)^{1/m}} \tag{5}$$

then there is a tree of height $D := Lm$ on which the strategy needs at least $L\binom{m}{2} = \Omega((n/k + D) \cdot \log k / \log\log k)$ rounds. To complete the proof, we need to show that (5) holds for all $1 \leq k \leq n \log^c n$. We split the analysis to two cases. Let us first assume $k \geq n/m$ and thus $L = 1$. This implies

$$\frac{n^{1+1/m}}{6L(m+1)^2(2L)^{1/m}} \geq \frac{n^{1+1/m}}{24m^2} \geq \frac{n \log^{8+c} n}{24 \log^2 n} \geq k,$$

when $k \leq n \log^c n$ and for sufficiently large n.

Now we consider the case $k < n/m$. Using that assumption and the definition of L we obtain $L(m+1)^2 \leq 4mn/k$ and $2L \leq 4n/(mk)$. Putting it all together we have

$$\frac{n^{1+1/m}}{6L(m+1)^2(2L)^{1/m}} = \frac{n}{6L(m+1)^2} \left(\frac{n}{2L}\right)^{1/m} \geq \frac{k}{24m} \left(\frac{mk}{4}\right)^{1/m} \geq k,$$

where the last inequality holds for $k \geq \sqrt{n}$ because, for sufficiently large n,

$$(mk)^{1/m} \geq k^{1/m} \geq e^{\frac{\log n}{2m}} \geq e^{\frac{(8+c)\log\log n}{4}} \geq (\log n)^2 \geq 100m.$$

□

Theorem 2. *Given any constant $\varepsilon > 0$ there is an integer $D = D(\varepsilon)$ such that for sufficiently large n and for every deterministic exploration strategy using $k \leq D \cdot n^{1+\varepsilon}$ agents, there exists a tree on n vertices and with height D on which the strategy needs at least $D/(5\varepsilon)$ rounds.*

Proof. We choose $L = 1$ and $m = \lceil 1/2\varepsilon \rceil$ in Lemma 1. The claim is trivial unless $\varepsilon < 1/5$, so we can eliminate rounding and assume generously that $1/m \geq 1.4\varepsilon$. The condition $n \geq L \cdot 16^m$ is clearly satisfied for sufficiently large n.

By Lemma 1, there is a tree T of height m that needs time $\binom{m}{2} \geq m/(5\varepsilon)$ to be explored, provided the team has size at most (for n sufficiently large)

$$k \leq n^{1+1.4\varepsilon}/(12(m+1)^2) \leq m \cdot n^{1+\varepsilon} = D \cdot n^{1+\varepsilon}.$$

□

Theorem 3. *For every n and every deterministic exploration strategy using $k = n$ agents, there exists a tree T on n vertices and with height $D = \omega(1)$ such that the strategy needs at least $D^2/3$ rounds to explore T.*

Proof. We choose $L = 1$ and $m = \lceil \sqrt{\log n} \rceil$ in Lemma 1. Then $n \geq L \cdot 16^m$ holds for sufficiently large n. Note also that for sufficiently large n,

$$\frac{n^{1+1/m}}{12(m+1)^2} = \Omega(n \cdot e^{\sqrt{\log n}}/\log n) \geq n.$$

The lemma now states that there exists a tree T of height m such that the given strategy with $k = n$ agents needs at least $\binom{m}{2}$ rounds to explore T. Since for large enough n we have $\binom{m}{2} \geq m^2/3$, this implies the theorem. □

Theorem 4. *For any function $D \leq n^{o(1)}$ and any exploration strategy using $k = \Theta(n)$ agents, the competitive ratio on the set of all trees of size n and height at least D is $\omega(1)$.*

Proof. Suppose that $D \leq n^{o(1)}$, i.e., $D = n^{1/f(n)}$, where $f(n)$ is a function which tends to infinity with n. Let $L = D$ and note that we have

$$\frac{16L^{1/m}}{n^{1/m}} \leq \frac{L^{1+1/m}(m+1)^2}{n^{1/m}} \leq \frac{4m^2 n^{2/f(n)}}{n^{1/m}}.$$

If we choose $m = m(n) = \omega(1)$ as a function growing sufficiently slowly such that we have $m \leq \min\{(f(n))^{1/2}, (\log n)^{1/2}\}$, then the following is true:

$$\frac{4m^2 n^{2/f(n)}}{n^{1/m}} = 4 \cdot e^{2\log m + 2(\log n)/f(n) - \log n/m} \to 0.$$

This implies $16L^{1/m} = o(n^{1/m})$ and $L^{1+1/m}(m+1)^2 = o(n^{1/m})$. In particular, $n \geq L \cdot 16^m$ for sufficiently large n. Moreover, if n is large enough then $k = \Theta(n)$ implies

$$\frac{n^{1+1/m}}{6L(m+1)^2(2L)^{1/m}} = \frac{n^{1+1/m}}{o(n^{1/m})} \geq k.$$

By Lemma 1, there exists a tree T with height $Lm \geq D$ on which the strategy needs $L\binom{m}{2} = \omega(Lm)$ rounds. Since $k = \Theta(n)$, the offline optimum is $\mathcal{O}(Lm + n/k) = \mathcal{O}(Lm)$, so the competitive ratio on the set of trees of height at least D is $\omega(1)$, as claimed. □

6 Conclusions

In this paper we presented new lower bounds for collaborative tree exploration. Including our results, the following bounds are now known. For $k = \mathcal{O}(1)$ or $k \geq D \cdot n^{1+\varepsilon}$ agents, a competitive ratio of $\Theta(1)$ can be achieved. For $\omega(1) \leq k \leq n \log^c n$, the best-possible competitive ratio is bounded by $\Omega(\log k/\log\log k)$, and no constant competitive ratio is possible when $n \log^c n \leq k \leq D \cdot n^{1+o(1)}$. On the other hand, the best exploration algorithms for trees in the domain $k \leq D \cdot n^{1+o(1)}$ stay close to the trivial competitive ratio of k (the best ratios are $k/\log k$ and $k^{o(1)}$, depending on the domain).

In summary, we now fully understand the domain where constant competitive ratios are possible, but, outside this domain, a wide gap persists.

Acknowledgments

We would like to thank Rajko Nenadov for useful discussions.

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM Journal on Computing 29(4), 1164–1188 (2000)
2. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovász, L., Rackoff, C.: Random walks, universal traversal sequences, and the complexity of maze problems. In: Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS). pp. 218–223 (1979)
3. Alon, N., Avin, C., Koucký, M., Kozma, G., Lotker, Z., Tuttle, M.R.: Many random walks are faster than one. Combinatorics, Probability and Computing 20(4), 481–502 (2011)
4. Ambühl, C., Gąsieniec, L., Pelc, A., Radzik, T., Zhang, X.: Tree exploration with logarithmic memory. ACM Transactions on Algorithms 7(2), 1–21 (2011)
5. Awerbuch, B., Betke, M., Rivest, R.L., Singh, M.: Piecemeal graph exploration by a mobile robot. Information and Computation 152(2), 155–172 (1999)
6. Bender, M.A., Fernández, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: Exploring and mapping directed graphs. Information and Computation 176(1), 1–21 (2002)
7. Bender, M.A., Slonim, D.K.: The power of team exploration: Two robots can learn unlabeled directed graphs. In: Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS). pp. 75–85 (1994)
8. Blum, M., Kozen, D.: On the power of the compass (or, why mazes are easier to search than graphs). In: Proceedings of the 19th Annual Symposium on Foundations of Computer Science (FOCS). pp. 132–142 (1978)
9. Chalopin, J., Das, S., Disser, Y., Mihalák, M., Widmayer, P.: Mapping simple polygons: How robots benefit from looking back. Algorithmica 65(1), 43–59 (2011)
10. Chalopin, J., Das, S., Disser, Y., Mihalák, M., Widmayer, P.: Mapping simple polygons. ACM Transactions on Algorithms 11(4), 1–16 (2015)
11. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph. Journal of Graph Theory 32(3), 265–297 (1999)
12. Dereniowski, D., Disser, Y., Kosowski, A., Pająk, D., Uznański, P.: Fast collaborative graph exploration. Information and Computation 243, 37–49 (2015)
13. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree exploration with little memory. Journal of Algorithms 51(1), 38–63 (2004)
14. Disser, Y., Hackfeld, J., Klimm, M.: Undirected graph exploration with $\Theta(\log \log n)$ pebbles. In: Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 25–39 (2016)
15. Duncan, C.A., Kobourov, S.G., Kumar, V.S.A.: Optimal constrained graph exploration. ACM Transactions on Algorithms pp. 380–402 (2006)
16. Dynia, M., Łopuszański, J., Schindelhauer, C.: Why robots need maps. In: Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity (SIROCCO). pp. 41–50 (2007)
17. Elsässer, R., Sauerwald, T.: Tight bounds for the cover time of multiple random walks. Theoretical Computer Science 412(24), 2623–2641 (2011)
18. Fraigniaud, P., Gąsieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks 48(3), 166–177 (2006)
19. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. Theoretical Computer Science 345(2-3), 331–344 (2005)
20. Higashikawa, Y., Katoh, N., Langerman, S., Tanigawa, S.i.: Online graph exploration algorithms for cycles and trees by multiple searchers. Journal of Combinatorial Optimization 28(2), 480–495 (2012)
21. Hoffmann, F.: One pebble does not suffice to search plane labyrinths. In: Proceedings of the 3rd International Symposium on Fundamentals of Computation Theory (FCT). pp. 433–444 (1981)
22. Ortolf, C., Schindelhauer, C.: Online multi-robot exploration of grid graphs with rectangular obstacles. In: Proceedinbgs of the 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). pp. 27–36 (2012)
23. Ortolf, C., Schindelhauer, C.: A recursive approach to multi-robot exploration of trees. In: Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO). pp. 343–354 (2014)
24. Ortolf, C., Schindelhauer, C.: Strategies for parallel unaware cleaners. Theoretical Computer Science 608, 178–189 (2015)
25. Reingold, O.: Undirected connectivity in log-space. Journal of the ACM 55(4), 1–24 (2008)