Two-Agent Tree Evacuation

Henri Devillez, Béni Egressy(✉), Robin Fritsch, and Roger Wattenhofer

ETH Zürich, Switzerland
{hdevillez,begressy,rfritsch,wattenhofer}@ethz.ch

Abstract. We study the problem of evacuating two agents from a tree graph, through an unknown exit located at one of the nodes. Initially, the agents are located at the same starting node; they explore the graph until one of them finds the exit through which they can evacuate. The task is to minimize the time it takes until *both* agents evacuate, for a *worst case* placement of the exit. We consider two communication models, *global communication* where the agents can communicate at any time, and *local communication* where the agents can only communicate if they are at the same node at the same time. We show that the problem is NP-hard in both cases. We then present a 4/3-approximation algorithm for global and a 3/2-approximation algorithm for local communication.

1 Introduction

Imagine that two robots are trapped in a building. They have a map of the building, but most exits marked on the map are in fact blocked. Their goal is to find an exit that is still available, and evacuate. Armed with the map they devise a strategy to find an exit and evacuate as quickly as possible. How should they divide the work of checking all the locations for an available exit? In what order should they explore the locations? Should they meet up at some predefined location to exchange information?

More formally, we consider a group of k agents whose task is to find an exit node in a weighted graph $G = (V, E)$ and evacuate all agents through this exit. The agents all start at common node r, which we call the root. In the beginning, they know the graph and the edge weights and that at least one exit exists at one of the nodes, but not the location of the exit(s). In this paper, we study the case of $k = 2$ agents, one single exit, the graphs we consider will be trees. We are interested in worst case analysis so we can assume a single exit without loss of generality. Moreover two agents on a tree is the most fundamental case, encompassing already the challenges, trade-offs and insights of the more general setting. We will distinguish between two communication models. In the *local* model agents can only explicitly exchange information when they are at the same node at the same time. In the *global* model agents can exchange information at any time independent of their locations. Global communication is equivalent to a single central algorithm controlling all the agents.

The objective will be to minimize the time for *all* agents to evacuate the graph. In other words we minimize the last exit time, or *evacuation time*. We can view this problem as a two player game, where one player defines the exploration-evacuation strategy for all the agents and the second player, the adversary, then chooses the worst possible exit location for this strategy.

An optimum algorithm is one that achieves the minimum worst case evacuation time. We first show that minimizing the worst case evacuation time for two agents in a tree is NP-hard. This motivates the search for approximation algorithms. Their performance is measured by the approximation ratio: The ratio of the worst case evacuation time of the given algorithm and the worst case evacuation time of an optimum algorithm.

We start our search in the global communication setting and show that a simple bi-directional depth-first search (DFS) strategy has an approximation ratio of 7/5. As our first major result, we then present the *Longest Path Global Algorithm* and show that it has a tight approximation ratio of 4/3. We then focus on the local communication setting and start again by showing that the simple bi-directional approach gives an approximation ratio of 2. As our second major result, we strengthen this result with a centroid-based algorithm, proving a tight approximation ratio of 3/2. Finally we ask how much worse local communication can be compared to global communication on the same graph and bound the worst case ratio of the two models between 4/3 and 3/2. We finish with concluding remarks, pointing the reader to a whole range of open questions for future work.

2 Related Work

One of the most fundamental classes of problems in the context of mobile agents are search problems. These include a whole range of related problems, such as ants searching for food [16, 17, 22], graph exploration [10, 18, 19], rendezvous problems [7, 12, 13, 14, 20], patrolling robots [26, 27], and pursuit-evasion games [5, 6, 23]. Another example are graph evacuation problems, where one or more agents search for an exit, through which they can evacuate. Evacuation problems are of two main types; geometrical problems, where agents need to evacuate some shape, such as a triangle, a square or a disk [21, 25, 29], and evacuation problems on graphs. The latter type has not been explored to a great extent; the problem has been considered on lines [1, 3] and so-called m-rays [8, 9, 30], but not on general graphs. Borowiecki et al. [24] consider the problem of evacuating multiple agents from distinct nodes using multiple exits, but this problem is very different from ours with a focus on avoiding congestion and bottlenecks.

Another related problem is swarm exploration [15]. A swarm of mobile agents starting at the root of a tree has to visit every node, whilst staying within a distance of d, where d is called the range of the swarm. Unlike swarm exploration, graph evacuation does not put a hard constraint on the distance between agents, the agents stay close only when it is optimal.

To the best of our knowledge, our work is the first to consider evacuating two agents from a tree or graph through a single unknown exit. Although the geometric setting already leads to elaborate strategies and interesting insights [25, 29], graphs are a much more general and more widely applicable setting for multi-agent evacuation.

A related graph exploration problem is the multiple Traveling Salesmen Problem (mTSP) [11]. The mTSP is a generalization of the well-known traveling salesman problem (TSP), where more than one salesman is allowed to be used in the solution. Unlike mTSP, in graph evacuation one also has to consider how far apart the agents get during the exploration, as they eventually all have to converge to the exit.

3 Preliminaries

We start with an example to illustrate some important aspects of the problem.

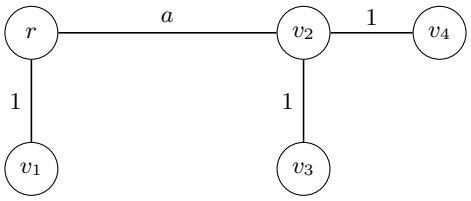


Fig. 1: An instance of tree evacuation. The agents start at r and $d(r, v_2) = a > 0$.

Take a look at Figure 1. Let's first consider local communication. We observe that there are two competing strategies that could be optimal, depending on the value of a. The first is for the agents to stay together and explore everything together starting with v_1, then v_2, v_3, and finally v_4. By staying together, they can both evacuate immediately as soon as they find an exit. On the other hand the exploration is not parallelized and as such is not very efficient. The worst case is when the exit is located at v_4 giving an evacuation time of $a + 5$.

The second strategy is for the agents to split up at the beginning. The agents explore v_1 and v_3 in parallel, meeting up at v_2 to share information at time $t = a + 2$. The agents now know exactly where the exit is and can head straight there. The worst case is when the exit is at v_1 giving a total time of $2a + 3$. This strategy demonstrates the importance of the agents meeting to share information in the local communication setting.

Which strategy is better depends on the value of a. This example illuminates the two key aspects of the problem: to explore the graph efficiently in parallel, and to stay close enough in case the exit is found. A good algorithm has to balance these two (orthogonal) goals.

Let us now consider the same example with global communication to make another interesting observation. Let $a = 4$. We notice that the optimum strategy

depends on whether the agents can abort traversing an edge when they are part way across. If they cannot, then splitting up would give a worst case evacuation time of $2a + 1 = 9$ with the exit at v_1, and staying together would also give a worst case evacuation time of $a+5 = 9$. However if they can abort traversing the long edge, then the strategy of splitting up gives the optimum evacuation time 7. We will assume throughout the rest of the paper that agents can traverse an edge part way.

We start by showing that it is NP-hard to find the optimal strategy on trees.

Lemma 1. *Consider weighted tree evacuation with $k = 2$ agents. Finding a strategy to minimize the worst case evacuation time is NP-hard.*

Proof. We show this by reduction from PARTITION [2]. Let $S = (a_1, a_2, \ldots, a_n)$ be a multiset of positive integers, i.e., an instance of partition with $\sum_i a_i = 2M$. We construct the following star graph:

- Add root node r
- For each $a_i \in S$, we add node v_i and edge $e_i = (r, v_i)$ with weight $w(e_i) = a_i$
- In addition we add node v_{n+1} and edge $e_{n+1} = (r, v_{n+1})$ with weight $w(e_{n+1}) = 2M$

Let r be the starting node for the agents. Then this gives an instance of graph evacuation. We claim that there is a strategy with worst case evacuation time $4M$ if and only if there is a solution to the partition problem.

For the simple direction, if there is a solution to the partition problem, then the agents can explore nodes $\{r, v_1, \ldots, v_n\}$ and return to r in time $2M$. If either agent finds the exit, say at v_i, then the agents can evacuate in additional time $a_i \leq 2M$, giving at most $2M + a_i \leq 4M$ in total. On the other hand if the exit is at v_{n+1}, since the agents have already checked all the other vertices at time $2M$, they can go straight to v_{n+1} together and exit in total time $2M + 2M$.

Now suppose there is no solution to the partition problem. The agents can still choose to explore all nodes $\{r, v_1, \ldots, v_n\}$ first, leaving v_{n+1} till last. So first suppose node v_{n+1} is explored last. Then the adversary places the exit at v_{n+1}. Since there is no solution to the partition problem, exploring the other nodes and returning to r requires time strictly greater than $2M$ in total. So one of the agents will require strictly more than $2M + 2M = 4M$ to reach the exit. On the other hand, suppose the agents visit node v_{n+1} before exploring one of the nodes, say v_j. Then the adversary places the exit at v_j and the agent visiting v_{n+1} has to return, requiring at least $4M + a_j \geq 4M + 1$ to reach the exit at v_j.

Thus, deciding whether the optimum worst case evacuation time is greater than $4M$ is as hard as partition, so tree evacuation with 2 agents is NP-hard. □

Note that we do not use anywhere in the proof that the communication model is local or global. Note also that the proof can easily be extended to $k > 2$ agents, for example by simply adding $k - 2$ edges of length M.

Since finding the optimal strategy is NP-hard we will be looking for approximation algorithms. We use the following general lower bound on the cost OPT of the optimal algorithm to prove these approximation ratios.

Lemma 2. *Let TSP be the length of the shortest traversal of all vertices of a graph $G = (V, E)$, i.e. the solution to the traveling salesperson problem. Then $OPT \geq TSP/2$.*

Proof. For any order in which the agents explore the vertices, the adversary can choose the last node they explore to be the exit. That means, the two agents must together visit all vertices. Moreover, since they start and finish at common vertices (the root and the exit), they together perform a complete traversal of the graph. The length of the shortest traversal is TSP, so at least one of the agents must travel a distance of at least $TSP/2$, implying $OPT \geq TSP/2$. □

This lemma applies to general graphs and any communication model. Since the shortest traversal of a tree graph uses every edge exactly twice, we get the following corollary for trees.

Corollary 3. *For a tree, $OPT \geq W$ where W is the total weight of the tree.*

Now we can prove an algorithm has an approximation ratio of at most c by showing that it always requires at most cW time to evacuate the agents.

Observation 4. *The best approximation ratio we can hope for using only this lower bound, $OPT \geq W$, is $3/2$ for either communication model.*

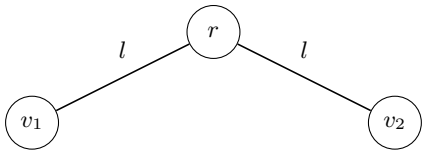


Fig. 2: Example of a tree graph with optimum worst case evacuation time $OPT = \frac{3}{2}W$. If the agents split up and reach v_1 and v_2 at the same time, then one of the agents will exit at time $3l$. If the agents do not split, then whichever leaf they explore first, the adversary can place the exit at the other, so that the agents again need $3l$ time to evacuate. Note also that $W = 2l$.

See Figure 2 for an example of a tree with $OPT = \frac{3}{2}W$. Clearly no algorithm can do better than this, so we cannot get a better approximation ratio than $3/2$ if we rely solely on the lower bound $OPT \geq W$. To prove a better approximation ratio, we would need a tighter analysis between the approximation algorithm and an optimum algorithm. Therefore, motivated by Observation 4, we give another lower bound for OPT.

Lemma 5. *Let v_1, v_2 be any two nodes of a tree rooted at r, with $d(r, v_1) \geq d(r, v_2)$. And let P_i denote the path from r to node v_i. Then $OPT \geq d(r, v_1) + 2d(v_2, P_1)$, where $d(v_2, P_1) = \min_{u \in P_1} d(v_2, u)$ is the shortest distance between v_2 and any node on the path P_1.*

Proof. Note that the adversary can force the two agents to explore all of the vertices in the graph, in particular both v_1 and v_2. Of course the agents can split up, whereby they visit v_1 and v_2 in parallel, but the adversary can place the exit at the node explored later by the agents. Therefore at least one of the agents visits both nodes. This gives a lower bound for OPT of

$$\begin{aligned} OPT &\geq \min\{d(r, v_1) + d(v_1, v_2), d(r, v_2) + d(v_2, v_1)\} \\ &= d(r, v_2) + d(v_2, v_1) \\ &= d(r, v_1) + 2 \min_{u \in P_1} d(v_2, u) \\ &= d(r, v_1) + 2d(v_2, P_1) \end{aligned}$$

□

In particular, applying Lemma 5 to v_1 and v_2 in Figure 2 gives us a tight lower bound for OPT. The lemma gives $OPT \geq 3l$, and $3l$ is indeed the optimum.

4 Global Communication

In the global setting, a simple way to evacuate a general graph is for the two agents to follow a traversal of the graph in opposite directions.

Algorithm 1: Bi-directional traversal (BiT)

input : A graph G with 2 agents at node r
output: The agents explore and evacuate the graph

```
1 Find a shortest traversal R of G
2 while the exit is not found do
3 |   The two agents traverse R in opposite directions starting from r
4 Both agents go directly to the exit
```

Lemma 6. *Algorithm 1 (BiT) has an approximation ratio of at most* 3/2 *for* 2*-agent global evacuation on trees.*

Proof. We can find the shortest traversal R of a tree in linear time using depth first search (DFS). Let $w(R)$ be the total length of this traversal. But for trees we know that $w(R) = 2W$. Let $d_R(t)$ be the distance between the two agents after time t for $0 \leq t \leq w(R)/2 = W$ if they simply follow the traversal and ignore the exit.

Then the worst case cost of the algorithm is $\max_{0 \leq t \leq W}\{t + d_R(t)\}$ since one of the agents needs to travel the distance $d_R(t)$ to the exit if the other finds it at time t. The agents start in the same place, and will also finish in the same place halfway around the traversal if they do not find the exit on their way. Hence, $d_R(0) = d_R(W) = 0$. Since the agents can only distance/near themselves at a rate of 2 units per time, their distance at time t is bounded by $2\min\{t, W - t\}$. With this we conclude

$$\max_{0 \leq t \leq W}\{t + d_R(t)\} \leq \max_{0 \leq t \leq W}\{t + 2\min\{t, W - t\}\} = \frac{3}{2}W \leq \frac{3}{2}OPT$$

where equality is attained at $t = W/2$ and the last inequality follows from Corollary 3. □

In principle Lemma 6 also holds for general graphs. However, finding the shortest traversal of a graph is NP-hard in general. Instead we can alter BiT to use a polynomial time approximation for the shortest traversal in line 1 of the algorithm.

Corollary 7. *BiT using the Christofides approximation for TSP, has an approximation ratio of at most* $9/4$ *for* 2*-agent global evacuation on general graphs.*

Proof. Let C be the length of the traversal of the graph found by Christofides [4]. If the two agents follow this traversal, the same arguments as in the previous proof imply that

$$BiT \leq \frac{3}{4}C \leq \frac{3}{4} \cdot \frac{3}{2}TSP \leq \frac{9}{4}OPT.$$

□

We now give a tight analysis of the BiT for trees by combining the lower bounds on OPT from Corollary 3 and Lemma 5. This analysis cannot be used to improve the approximation ratio for general graphs in Corollary 7, because Lemma 5 does not hold in general graphs.

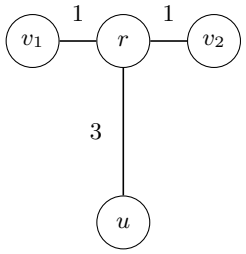
Lemma 8. *Algorithm 1 (BiT) has an approximation ratio of exactly* $7/5$ *for* 2*-agent global evacuation on trees.*

Proof. Figure 3(a) shows that the approximation ratio is no smaller than $7/5$.

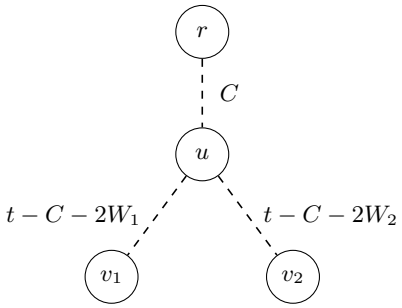
For the upper bound, consider a tree rooted at r. Consider the worst case location of the exit and let t be the time one of the agents reaches it. We can assume that all nodes of the graph have been visited at time t by the following argument. If this is not the case, we can simply remove all unexplored nodes from the graph, and the remaining graph will still be a connected tree. (If the other agent is on an edge at time t, we add a new node at that position, before removing unexplored nodes.) This will not change the cost of BiT since it can still choose the same exploration paths up to time t and the agents will still need to take the same paths to the exit. On the other hand, OPT will clearly not be increased by removing nodes, but might be decreased. This means the ratio BiT/OPT will certainly not decrease (and may well increase).

Let v_1 and v_2 be the positions of the two agents at time t. Up to time t, the first agent has traversed all edges of the path between the root r and v_1 once and a set of further edges twice. The same is true for the second agent and the path from r to v_2. Let W_1 and W_2 be the total weights of all edges (or parts of edges) that have been traversed twice by the first and second agent respectively.

Since no edge of the graph is traversed more than twice in total, the only overlap the two agents can have is a path from the root to a node u (where u could equal r). Let C be the length of this common path, see Figure 3(b).



(a) Lower bound example for BiT: The optimal strategy with a worst case cost of 5 is for the agents to explore v_1 and v_2 in parallel, and then head together to u. However, BiT could send one agent to u and the other to v_1 and v_2, giving an evacuation time of 7 if the exit is at u.



(b) Proof of the upper bound for BiT: The two agents start at r and both visit u at some point. At time t they are at v_1 and v_2, respectively. The dashed lines illustrate paths between vertices (rather than just edges).

Fig. 3

Since agent 1 arrives at v_1 at time t, the path from u to v_1 must have length $t-C-2W_1$. Similarly, the path from u to v_2 has length $t-C-2W_2$. Therefore, the total weight of the graph equals

$$W = C + W_1 + W_2 + (t - C - 2W_1) + (t - C - 2W_2) = 2t - (C + W_1 + W_2)$$

because we assume the whole graph is explored at time t. Thus,

$$\begin{aligned} BiT &= t + (t - C - 2W_1) + (t - C - 2W_2) \\ &= 3t - 2(C + W_1 + W_2) \\ &= \frac{3}{2}W - \frac{1}{2}(C + W_1 + W_2). \end{aligned}$$

If $C + W_1 + W_2 \geq W/5$, the cost of BiT is no larger than $\frac{7}{5}W$ which in turn is no larger than $\frac{7}{5}OPT$ by Corollary 3. Otherwise, applying Lemma 5 to v_1 and v_2, where without loss of generality $d(r, v_1) \geq d(r, v_2)$, yields

$$\begin{aligned} OPT &\geq C + (t - C - 2W_1) + 2(t - C - 2W_2) \\ &\geq 3t - 4(C + W_1 + W_2) \\ &= \frac{3}{2}W - \frac{5}{2}(C + W_1 + W_2). \end{aligned}$$

This implies

$$\frac{BiT}{OPT} \leq \frac{\frac{3}{2}W - \frac{1}{2}(C + W_1 + W_2)}{\frac{3}{2}W - \frac{5}{2}(C + W_1 + W_2)} \leq \frac{\frac{3}{2}W - \frac{1}{2}\frac{W}{5}}{\frac{3}{2}W - \frac{5}{2}\frac{W}{5}} = \frac{7}{5}. \qquad \square$$

To achieve a better approximation ratio, we will use what we call a *longest path approach*. Given a tree, let the path P_L from root r to v_L be a longest path starting at r and let L denote its length. In a longest path approach, both agents explore the tree sequentially along P_L. In other words, an agent passes through each edge of P_L at most once during exploration, making excursions to explore subtrees along the way. Note that the longest path approach is the optimal strategy for a single agent to explore a tree.

Let E' be the set of all edges not in P_L, with weight $w(E') = W - L$. The edges in E' form subtrees rooted along the path P_L. We would like each agent to explore half of E'. Let w be the walk from r to v_L of length $2W - L$ that traverses every edge on P_L once and all other edges twice (via a DFS along each subtree). Furthermore, let w' be the (non-connected) walk $w \setminus P_L$ of length $2(W - L)$. Let p be the midpoint of w'.

Algorithm 2: Longest Path Algorithm (LPA)

input : A tree G with 2 agents at its root r, the longest path P_L, the walk w and the point p as defined previously
output: The agents explore and evacuate the graph

```
while the exit is not found do
    The two agents explore the graph in parallel
    Agent 1 traverses w up to point p, then takes the shortest path from p to v_L
    Agent 2 takes the shortest path from r to p and then traverses the remaining part of w to v_L
Both agents go directly to the exit
```

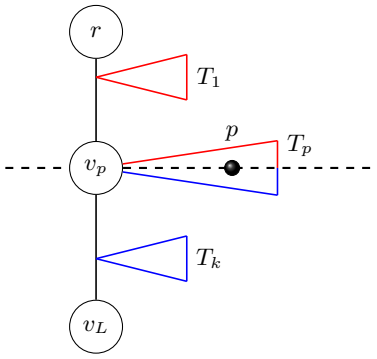
Following this algorithm, the agents arrive at v_L at the same time (if the exit is not found earlier), since they both traverse P_L, half of w' and the shortest path from p to P_L. In particular, the only part of E' they both explore is the shortest path from p to P_L.

Lemma 9. *If the two agents follow a longest path approach, then*

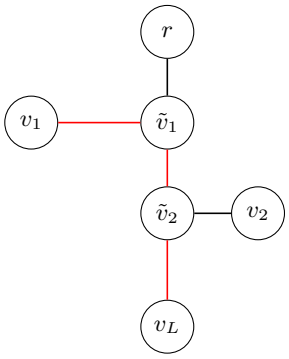
$$max\{d(v_1, v_L), d(v_2, v_L)\} \geq d(v_1, v_2)$$

at all times, where v_1 and v_2 are the positions of the agents at some time during exploration.

Proof. Let $\tilde{v}_i$ be the the furthest node along P_L that agent i has reached. Without loss of generality, we assume that $\tilde{v}_1$ is closer to the root than $\tilde{v}_2$. See Figure 4(b). Thus, the distance between the agents is $d(v_1, \tilde{v}_1) + d(\tilde{v}_1, \tilde{v}_2) + d(\tilde{v}_2, v_2)$ while the distance between agent 1 and v_L is $d(v_1, v_L) = d(v_1, \tilde{v}_1) + d(\tilde{v}_1, \tilde{v}_2) + d(\tilde{v}_2, v_L)$. We observe that $d(\tilde{v}_2, v_2)$ cannot be greater than $d(\tilde{v}_2, v_L)$, as otherwise v_2 would be further away from the root than v_L. We conclude that $max(d(v_1, v_L), d(v_2, v_L)) \geq d(v_1, v_2)$. □



(a) The longest path algorithm (LPA). The longest path P_L is the path from r to v_L. T_p is the subtree containing point p as described in the algorithm, and v_p is the root of T_p on P_L. Agent 1 explores the red edges and agent 2 the blue edges.



(b) An upper bound (red) on $d(v_1, v_2)$ for a longest path algorithm. The node v_L is the furthest node from r, and v_1 and v_2 are the locations of the agents.

Fig. 4

Theorem 10. *Algorithm 2 (LPA) has an approximation ratio of exactly* 4/3 *for* 2*-agent global evacuation on trees.*

Proof. We defer the lower bound for the approximation ratio to Observation 11. In the following we prove the upper bound.

Let LPA be the worst case evacuation time of the LPA algorithm. Without loss of generality assume the exit is found by agent 1 at a node v_1 at time t_1 and that the second agent is at position v_2 at this moment. The evacuation time is then $t_1 + d(v_1, v_2)$. But if the adversary had placed the exit at v_L, then by Lemma 9 the evacuation time would have been at least $t_1 + max\{d(v_1, v_L), d(v_2, v_L)\} \geq t_1 + d(v_1, v_2)$. Therefore the worst case evacuation time occurs when the exit is at v_L. The agents reach v_L at the same time at

$$LPA = d(r, v_L) + \frac{2(W - L)}{2} + d(p, v_p) = W + d(p, v_p)$$

where p is as described in the algorithm and v_p is the closest node on P_L to p, in other words the root of the subtree containing p.

Let $L_2 = d(p, v_p) \leq L$. Applying Lemma 5 to p and v_L gives a lower bound of $OPT \geq L + 2L_2$. Combining this with the lower bound from Corollary 3 gives us the approximation ratio:

$$\frac{LPA}{OPT} \leq \frac{W + L_2}{\max\{W, L + 2L_2\}}$$

If $L_2 \leq W/3$, then we get the result with the lower bound of W. On the other hand if $L_2 > W/3$, then $W + L_2 < 4L_2$ and since $L \geq L_2$, also $L + 2L_2 \geq 3L_2$; thus we have $\frac{LPA}{OPT} < \frac{4L_2}{3L_2} = 4/3$. □

Observation 11. *4/3 is a lower bound for the approximation ratio of any algorithm based on the longest path approach.*

See Figure 5 for an example of a tree where any algorithm visiting the furthest leaf last can only achieve an approximation ratio of at least 4/3. Therefore LPA is a best possible algorithm based on the longest path approach and the analysis in Theorem 10 is tight.

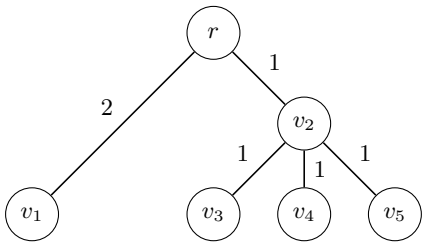


Fig. 5: Example of a tree where any algorithm visiting the furthest leaf last can achieve an approximation ratio of at best 4/3. If the agents explore v_1 last, then they first explore in the direction of v_2; they can visit v_3 and v_4 respectively, returning to v_2 at time 3. The exit can still be at v_1 or v_5 so the best the agents can do is split up (or explore v_5 and then v_1 together), but with the exit at v_1, the agent(s) exploring v_5 will only reach the exit at time 8. On the other hand, an optimum strategy would be for agents 1 and 2 to explore v_1 and v_3 respectively; if the exit is found, they can evacuate in time 6, and otherwise agent 2 continues by exploring v_4, before the agents meet at v_2 at $t = 5$; and now the exit can be reached in 1 step. This gives an approximation ratio of $8/6 = 4/3$.

5 Local Communication

We turn our attention to the local communication model. Local communication is of course weaker than global communication and indeed there are graphs, where agents with local communication need significantly more time to evacuate in the worst case. We will show such examples at the end of the section, but first we present approximation algorithms for this communication model.

Observation 12. *Depth-first search (DFS) has an approximation ratio of* 2 *for* 2*-agent local evacuation on trees.*

This observation is immediate. The length of a DFS traversal is $2W$, so if the agents walk together along the DFS route, they will evacuate in time at most $2W$. This can be improved slightly by ending the traversal at v_L, giving $2W - L$.

Next we propose an algorithm to improve this approximation ratio to 3/2. We start with some definitions, before giving an outline of the algorithm.

Definition 13 (child subtree, centroid). *Given a tree G and a node v, a child subtree of v is defined as the connected subtree of G containing v and a connected component of $G \setminus \{v\}$. A node with k neighbours has k child subtrees.*

Given a tree G and a point p on edge (u, v), a child subtree of p is defined as above by treating p as a node of the graph. In other words, a child subtree of p is defined as the connected subtree of G containing p and a connected component of G without the edge (u, v). A point p on an edge has 2 child subtrees. Note that both child subtrees only contain part of the edge (u, v).

Given a weighted tree G with total weight W, the centroid of G is defined as the vertex or point c, such that all child subtrees of c have weight at most $W/2$.

We include a DFS-like algorithm for finding the centroid of a tree in Appendix A.1. Note that the centroid is unique for trees with positive edge weights, but this uniqueness is not needed in the algorithm. We can now give an outline of our algorithm for 2-agent local tree evacuation.

A key ingredient of the algorithm is that we use the centroid of the tree (c) as a meeting point in each iteration. The advantage of meeting at c is that every node of the graph is at distance at most $W/2$. So in particular if either agent finds the exit, then after meeting at c they can evacuate together in additional time at most $W/2$. This means we can safely assign a budget of W to each agent for exploring the child subtrees of c, see Figure 7. Taking these budgets into account, the algorithm assigns child subtrees to the agents. We call the union of the unassigned child subtrees T. Note that T is a (connected) tree rooted at c.

Another key ingredient of the algorithm is that the agent with more leftover budget, can use its extra budget to start exploring T with a careful subroutine, which ensures that the unexplored edges form a *connected* subtree whilst guaranteeing a certain amount of further exploration. We start by presenting this subroutine and then we present the full algorithm.

Lemma 14. *Algorithm 3 is correct. That is, it always terminates, the agent traverses a subtree of G rooted at r in total time at most B, and the agent leaves an unexplored connected subtree rooted at r_1 with total weight at most $W - \Delta$ and $d(r, r_1) \leq \Delta$.*

See Figure 6(a) for an illustration of Algorithm 3 and see Figure 6(b) for an example showing that the analysis in Lemma 14 is tight.

Proof. We prove that before each recursive call of the algorithm, G' is an unexplored connected tree and its weight has decreased by at least the same amount as the decrease in $\Delta' = B' - W'$, where $W' = w(G')$. In addition, we have to show the agent's movements never exceed the budget B'. By recursively calling the algorithm until termination, this proves the lemma.

If r has a single child further away than Δ, then we are done since the agent can simply explore up to distance Δ along the edge and then return to r. This

Algorithm 3: Single Agent Lightest Subtree Algorithm (SALSA)

input : A tree G rooted at r with total weight $W = w(G)$
An agent with an exploration budget $B = W + \Delta$, $\Delta \leq W$
output: Agent traverses a subtree of G in time at most B, starting and finishing at r. It leaves an unexplored connected subtree rooted at r_1 (possibly on an edge) with weight at most $W - \Delta$ and $d(r, r_1) \leq \Delta$.

1 **if** $B > w(G)$ **then**
2 **if** *r has a single child* **then**
3 **if** $d(r, r.child) \geq \Delta$ **then**
4 Move Δ units along the edge $e(r, r.child)$ and place r' here
5 $B' = B - 2\Delta$
6 $G' = G \setminus \{r\} \cup \{r'\}$
7 **else**
8 $r' = r.child$
9 Go to r'
10 $B' = B - 2d(r, r')$
11 $G' = G \setminus \{r\}$
12 **else**
13 Traverse lightest (lowest total weight) unexplored child subtree T_1 of r with DFS
14 $r' = r$
15 $B' = B - 2w(T_1)$
16 $G' = G \setminus T_1 \cup \{r\}$
17 Call SALSA(G', r', B')
18 Go back to r
19 **else**
20 $r_1 = r$

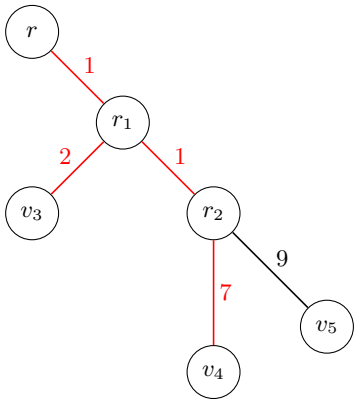
leaves an unexplored connected subtree rooted at r' as required. Clearly the budget is enough to do this and go back to r, since $B = W + \Delta \geq 2\Delta$.

If r has a single child closer than Δ, then the agent can move to the child at a cost of $d := d(r, r.child)$. The agent will incur the same cost again when returning to r at the end so we decrease the budget by $2d$. Note that G' is still an unexplored connected subtree. Moreover the weight of G' is d smaller than the weight of G at the start of the call. So we have

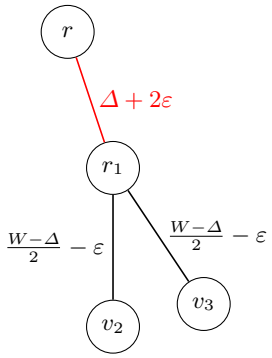
$$W' = w(G') = W - d$$
$$B' = B - 2d = W + \Delta - 2d = W' + (\Delta - d)$$

which implies $\Delta' = \Delta - d$.

For the remaining cases, we use the fact that a node r can have at most 1 child subtree of total weight greater than $\frac{1}{2}(W + \Delta)$. This is clear, else G would have weight greater than $W + \Delta \geq W$, which is not possible. Therefore we always have enough budget in line 13 of the algorithm to explore the lightest unexplored child subtree T_1. Since we explore a whole child subtree T_1 and leave all other child subtrees unexplored, G' is still an unexplored connected subtree.



(a) Illustration of SALSA. Let $W = 20$ and $\Delta = 5$. Budget $B = 20 + 5$. First, the agent goes to r_1, then v_3, then r_2 and has budget $B' = 25 - 8 = 17$ left. The unexplored graph has weight $w(G') = 16 < B'$, so the agent continues by exploring the lightest child subtree, v_4. Since $B' = 17 \geq 2(7)$, this is fine. After this, the agent does not have enough budget to explore further. Total budget used is $2(11) = 22 \leq 25$ and the agent leaves an unexplored subtree of weight 9. This is less than $W - \Delta = 20 - 5$, as claimed in Lemma 14.



(b) Worst case scenario showing that $W - \Delta$ is tight for the weight of the unexplored subtree. Budget is $B = W + \Delta$. After moving to r_1, the agent has budget $B' = W + \Delta - 2(\Delta + 2\varepsilon) < 2\left(\frac{W-\Delta}{2} - \varepsilon\right)$ left, so it can explore neither v_2 nor v_3. Instead the agent returns to r and r_1 becomes the new root.

Fig. 6

The weight of G' has decreased by $w(T_1)$ at a cost of $2w(T_1)$ budget, and the agent does not have to traverse any of T_1 on the way back to r at the end. So with $d = w(T_1)$ we have the same equations as above.

Finally note that when we explore a whole child subtree, then we do not increase $d(r, r')$, but we do decrease Δ'; and when we explore an edge, then we increase $d(r, r')$ by the same amount as we decrease Δ'. Therefore we ensure that we always have $d(r, r') \leq \Delta$. □

Theorem 15. *Algorithm 4 (CMA) has an approximation ratio of exactly* 3/2 *for* 2*-agent local evacuation on trees.*

Proof. For the lower bound on the approximation ratio see Appendix A.2. Similarly, an algorithm for finding the centroid can be found in Appendix A.1.

For the upper bound we show that in each iteration of Algorithm 4 the agents explore the graph with a time to weight ratio of at most $\frac{3}{2}$. Moreover after every

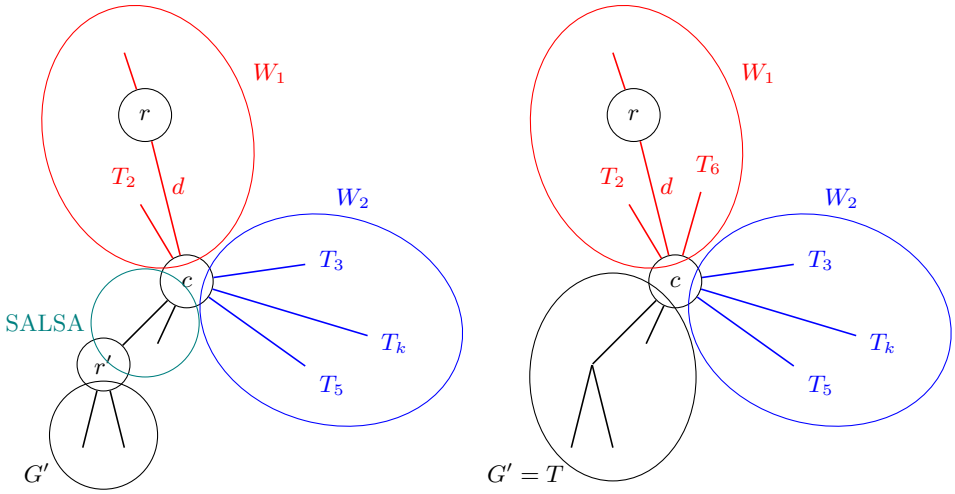
Algorithm 4: Centroid Meeting Algorithm (CMA)

input : A tree G with 2 agents at its root r with total weight W
output: The agents explore and evacuate the graph

$G' = G$, $r' = r$
while *the exit is not found* **do**
  Find the centroid of G' and call it c. Let $d = d(r', c)$
  Agent 1 gets a budget of $B_1 = w(G')$
  Agent 2 gets a budget of $B_2 = w(G')$
  We assign the child subtrees of c to the agents as follows
  begin
    We assign the child subtree $T_{r'}$ containing r' to agent 1 and decrease the agent's budget by $2w(T_{r'}) - d$ (if $r' = c$, we choose $T_{r'}$ to be an empty set)
    While agent 1's budget allows, we assign the unassigned child subtrees of c to agent 1 in decreasing order of weight, always at cost 2 times the weight
    We decrease agent 2's budget by d (cost to get to c)
    While agent 2's budget allows, we assign the unassigned child subtrees of c to agent 2 in decreasing order of weight, always at cost 2 times the weight
    Let T be the union of the remaining unassigned subtrees
  Agents 1 and 2 explore their assigned subtrees in parallel, finishing at c
  Let b be the first agent to arrive at c at time t_b, let a be the second agent to arrive at t_a and let $B = t_a - t_b$
  if $B > w(T)$ **then**
    Agent b starts to explore T using SALSA with budget B (while agent a finishes)
    $G', r' =$ unexplored connected subtree and its root returned by SALSA
    The agents meet at c and go to r'
  else
    Agent b waits for agent a at c
    $G' = T$, $r' = c$
The agents meet at c and go directly to the exit

iteration the agents are left with an unexplored connected tree G', rooted at r', with both agents located at r'. And in the final iteration, when the agents evacuate, they find the exit and evacuate in time at most $\frac{3}{2}w(G')$. Combining these claims, we can conclude that the agents find the exit and evacuate in time at most $\frac{3}{2}W$. Together with Corollary 3 this proves the theorem.

We denote the child subtrees of the centroid by $\{T_{r'}, T_1, T_2, \ldots, T_k\}$, where the degree of the centroid is $k+1$ and $T_{r'}$ is the subtree containing the starting node r'. If $r' = c$, $T_{r'}$ is the empty set and c has degree k. The child subtrees T_1 to T_k are ordered by increasing weight, i.e, $w(T_i) \geq w(T_{i-1})$ for all $i \geq 2$. Note that by definition of the centroid $w(T_i) \leq W/2$ for all $i = r', 1, \ldots, k$. The algorithm assigns $T_{r'}$ to agent 1. The algorithm then assigns each subtree in decreasing



(a) Case 1: $w(T) < B$ **(b)** Case 2: $w(T) \geq B$

Fig. 7: Illustration of the two cases in Algorithm 4. In case 1, the agent with more budget left explores T with SALSA. In case 2, the agents do not explore T in this iteration.

order of weight to the two agents, preferring always agent 1. Let W_1 and W_2 denote the total weights of subtrees assigned to agents 1 and 2 respectively.

Agent 1 can fully explore its subtrees and reach c at time $2W_1 - d$, where $d := d(r', c)$ is the distance from the starting node to the centroid. Agent 2 can fully explore its subtrees and reach c at $2W_2 + d$.

Claim: $W_1 + W_2 \geq W/2$

Assume the claim is not true for a contradiction, and $W_1 + W_2 < W/2$. Since $w(T_i) \leq W/2$ for all child subtrees, we must have at least two child subtrees T_1 and T_2, which cannot be explored by either agent. We can assume $w(T_1) \leq w(T_2)$. Then since $W_1 + W_2 + w(T_1) + w(T_2) \leq W$, we must have $2w(T_1) \leq W - (W_1 + W_2)$. And since T_1 does not fit into either agent's budget, we have the inequalities

$$2W_1 - d + (W - (W_1 + W_2)) \geq 2W_1 - d + 2w(T_1) > W$$
$$2W_2 + d + (W - (W_1 + W_2)) \geq 2W_2 + d + 2w(T_1) > W$$

which by adding up imply $2W > 2W$. Since this is a contradiction, we conclude that the claim is true.

Among the two agents, let agent a have a larger exploration time than agent b. Then we know the following about their respective exploration times t_a and t_b.

$$\begin{aligned} t_a &= \max\{2W_1 - d, 2W_2 + d\} \\ t_b &= \min\{2W_1 - d, 2W_2 + d\} \\ t_a + t_b &= 2(W_1 + W_2) \geq W \qquad (1) \\ t_a + t_b + 2w(T) &= 2W \qquad (2) \end{aligned}$$

We now consider the two cases from the algorithm (see Figure 7):

Case 1: $B > w(T)$

In this case agent b explores T using SALSA with budget $B = t_a - t_b$. We can apply Lemma 14 and we know that agent b can explore an additional weight of at least $\Delta = B - w(T) = t_a - t_b - w(T)$, leaving an unexplored subtree rooted at the new root r' (reassigned in line 17) at distance $d(c, r') \leq \Delta$.

Before the end of the iteration, the agents meet at the centroid c and then move to the new root r'. Using the above equations, we show that the iteration has efficient exploration with a time to weight ratio of at most $\frac{3}{2}$:

$$\begin{aligned} \text{total explored} &= \frac{t_a + t_b}{2} + (t_a - t_b - w(T)) \\ \text{time} &= t_a + (t_a - t_b - w(T)) \\ \frac{\text{time}}{\text{total explored}} &= \frac{4t_a - 2t_b - 2w(T)}{3t_a - t_b - 2w(T)} \overset{(2)}{=} \frac{5t_a - t_b - 2W}{4t_a - 2W} \overset{(1)}{\leq} \frac{6t_a - 3W}{4t_a - 2W} = \frac{3}{2} \end{aligned}$$

Case 2: $B \leq w(T)$

In this case we argue that even without any extra exploration by SALSA, the agents must have already explored efficiently with a time to weight ratio of at most $\frac{3}{2}$. By our claim, we have

$$t_a - t_b \leq w(T) \leq \frac{W}{2} \overset{(1)}{\leq} \frac{1}{2}(t_a + t_b).$$

Adding $t_a + t_b$ to the left and right sides we get $2t_a \leq 3(t_a + t_b)/2$. The agents have explored $(t_a + t_b)/2$ in time t_a giving

$$\frac{\text{time}}{\text{total explored}} = \frac{2t_a}{t_a + t_b} \leq \frac{3}{2}.$$

Thus we have shown that exploration is done with the required efficiency in each iteration. What remains to be shown is that in the final iteration, the agents find the exit and evacuate in time at most $\frac{3}{2}w(G')$. However this is clear from the choice of the meeting location c and the budgets $w(G')$. Since the agents meet at the centroid, any exit they find in the current iteration can be at a distance of at most $\frac{1}{2}w(G')$. And since they both arrive by $t = w(G')$, the evacuation time can be at most $\frac{3}{2}w(G')$ as required. □

We finish with a lower bound on the worst case ratio between two agents with global communication versus two agents with local communication. Clearly, agents with global communication can always do at least as well as agents with local communication. However, what about in the worst case? How much worse can local communication be? We return to Figure 5 to give an example where OPT_L is strictly larger than OPT_W. Recall that $OPT_W = 6$ and that any strategy visiting v_1 last would give an evacuation time of at least 8. Therefore with local communication we also explore v_1 and v_3 first. But now the agents have to meet at r to share the information; if they meet at v_2 it will be too late. But meeting at r leaves v_4 and v_5 unexplored, and the agents will have worst case evacuation time of 8, giving $OPT_L = 8$.

Together with Theorem 15 and Lemma 3, we bound the worst case ratio of local communication over global communication between 4/3 and 3/2.

6 Conclusion

To the best of our knowledge, our work is the first that considers evacuating two agents from a tree through a single unknown exit. We first show that the problem is NP-hard, motivating the search for approximation algorithms. Our main results are a 4/3-approximation algorithm for the global communication setting and a 3/2-approximation algorithm for the local communication setting.

The paper leaves a multitude of open questions for further research. One could search for better approximation algorithms to improve the approximation ratios. One could consider other communication models, such as blackboard communication [28] or communication with a limited radius. This paper focused on trees, but one could look at other graph classes. One could look at the case with more agents ($k > 2$) or different objectives, e.g., minimizing the average exit time of the agents. One could also consider the stochastic setting, where the exit is at a given node with some predetermined probability and the agents minimize the expected evacuation time. One could even look at a game theoretic scenario, where agents behave selfishly and do not go out of their way to inform the other agents about the exit location. How would this change the exploration strategy?

Acknowledgements: We would like to thank Nicolas Marxer and Tobias Zwahlen for fruitful discussions and the anonymous reviewers for their helpful comments.

References

[1] Anatole Beck. “On the linear search problem”. In: *Israel Journal of Mathematics* 2.4 (1964), pp. 221–228.

[2] Richard M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*. 1972, pp. 85–103.

[3] Shmuel Gal. "Minimax solutions for linear search problems". In: *SIAM Journal on Applied Mathematics* 27.1 (1974), pp. 17–30.
[4] Nicos Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem.* Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.
[5] Torrence D Parsons. "Pursuit-evasion in a graph". In: *Theory and applications of graphs*. Springer, 1978, pp. 426–441.
[6] Richard Nowakowski and Peter Winkler. "Vertex-to-vertex pursuit in a graph". In: *Discrete Mathematics* 43.2-3 (1983), pp. 235–239.
[7] Steve Alpern. "The rendezvous search problem". In: *SIAM Journal on Control and Optimization* 33.3 (1995), pp. 673–683.
[8] Ramzi F Hejazi, Tahir Husain, and Faisal I Khan. "Landfarming operation of oily sludge in arid region—human health risk assessment". In: *Journal of hazardous materials* 99.3 (2003), pp. 287–302.
[9] Alejandro López-Ortiz and Sven Schuierer. "On-line parallel heuristics, processor scheduling and robot searching under the competitive framework". In: *Theoretical Computer Science* 310.1-3 (2004), pp. 527–537.
[10] Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. "Graph exploration by a finite automaton". In: *Theoretical Computer Science* 345.2-3 (2005), pp. 331–344.
[11] Tolga Bektas. "The multiple traveling salesman problem: an overview of formulations and solution procedures". In: *omega* 34.3 (2006), pp. 209–219.
[12] Anders Dessmark, Pierre Fraigniaud, Dariusz R Kowalski, and Andrzej Pelc. "Deterministic rendezvous in graphs". In: *Algorithmica* 46.1 (2006), pp. 69–96.
[13] Evangelos Kranakis, Danny Krizanc, and Sergio Rajsbaum. "Mobile agent rendezvous: A survey". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2006, pp. 1–9.
[14] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. "Gathering few fat mobile robots in the plane". In: *Theoretical Computer Science* 410.6-7 (2009), pp. 481–499.
[15] Jurek Czyzowicz, Andrzej Pelc, and Mélanie Roy. "Tree exploration by a swarm of mobile agents". In: *International Conference On Principles Of Distributed Systems*. Springer. 2012, pp. 121–134.
[16] Ofer Feinerman and Amos Korman. "Memory lower bounds for randomized collaborative search and implications for biology". In: *International Symposium on Distributed Computing*. Springer. 2012, pp. 61–75.
[17] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sébastien Sereni. "Collaborative search on the plane without communication". In: *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. 2012, pp. 77–86.
[18] Klaus-Tycho Förster and Roger Wattenhofer. "Directed graph exploration". In: *International Conference On Principles Of Distributed Systems*. Springer. 2012, pp. 151–165.

[19] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. "Online graph exploration: New results on old and new algorithms". In: *Theoretical Computer Science* 463 (2012), pp. 62–72.

[20] Shantanu Das, Dariusz Dereniowski, Adrian Kosowski, and Przemysław Uznański. "Rendezvous of distance-aware mobile agents in unknown graphs". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2014, pp. 295–310.

[21] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil Shende. "Wireless autonomous robot evacuation from equilateral triangles and squares". In: *International Conference on Ad-Hoc Networks and Wireless*. Springer. 2015, pp. 181–194.

[22] Yuval Emek, Tobias Langner, David Stolz, Jara Uitto, and Roger Wattenhofer. "How many ants does it take to find the food?" In: *Theoretical Computer Science* 608 (2015), pp. 255–267.

[23] Klaus-Tycho Förster, Rijad Nuridini, Jara Uitto, and Roger Wattenhofer. "Lower bounds for the capture time: Linear, quadratic, and beyond". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2015, pp. 342–356.

[24] Piotr Borowiecki, Shantanu Das, Dariusz Dereniowski, and Łukasz Kuszner. "Distributed evacuation in graphs with multiple exits". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2016, pp. 228–241.

[25] Sebastian Brandt, Felix Laufenberg, Yuezhou Lv, David Stolz, and Roger Wattenhofer. "Collaboration without communication: Evacuating two robots from a disk". In: *International Conference on Algorithms and Complexity*. Springer. 2017, pp. 104–115.

[26] Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. "Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors)". In: *International Conference on Current Trends in Theory and Practice of Informatics*. Springer. 2017, pp. 229–240.

[27] Huda Chuangpishit, Jurek Czyzowicz, Leszek Gasieniec, Konstantinos Georgiou, Tomasz Jurdziński, and Evangelos Kranakis. "Patrolling a path connecting a set of points with unbalanced frequencies of visits". In: *International Conference on Current Trends in Theory and Practice of Informatics*. Springer. 2018, pp. 367–380.

[28] Ali Dorri, Salil S Kanhere, and Raja Jurdak. "Multi-agent systems: A survey". In: *Ieee Access* 6 (2018), pp. 28573–28593.

[29] Yann Disser and Sören Schmitt. "Evacuating two robots from a disk: a second cut". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2019, pp. 200–214.

[30] Sebastian Brandt, Klaus-Tycho Foerster, Benjamin Richner, and Roger Wattenhofer. "Wireless evacuation on m rays with k searchers". In: *Theoretical Computer Science* 811 (2020), pp. 56–69.

A Appendix

A.1 Centroid Algorithm

Algorithm 5: Algorithm for finding the Centroid of a tree

input : A tree T with total weight W
output: A point c (a vertex or a point on an edge), such that all child subtrees of c have weight at most $W/2$

```
1 Start at any node r
2 while some child subtree of r has weight greater than W/2 do
3 |   Let T_1 be the child subtree rooted at r with greatest weight, and let
  |    r_1 be the neighbour of r in this child subtree
4 |   if w(T_1) − d(r, r_1) < W/2 then
5 |   |   Place a node c on the edge (r, r_1) such that it splits T exactly
  |   |    into two child subtrees of equal weight
6 |   |   r = c
7 |   else
8 |   |   r = r_1
9 return r
```

We argue below that the algorithm is correct, that is, the algorithm finds a centroid. This also proves that every tree has a centroid.

Lemma 16. *Algorithm 5 terminates and returns a centroid.*

Proof. Consider an arbitrary iteration of the while loop. If the child subtree of r_1 containing r has a weight larger than $W/2$ (which is equivalent to the condition in line 4 being true), there exists a point on the edge (r, r_1) with two child subtrees of size exactly $W/2$ and the algorithm will return this. Otherwise, the algorithm moves its position along the edge (r, r_1). But the child subtree of r_1 containing r being smaller than $W/2$ also means that the algorithm cannot move along this edge in the other direction in any other iteration. So since the number of edges is bounded, the algorithm will terminate at some point. At this point the condition in line 2 will be false, meaning the algorithm returns a centroid. □

A.2 Lower Bound on competitive ratio of Algorithm 4 (CMA)

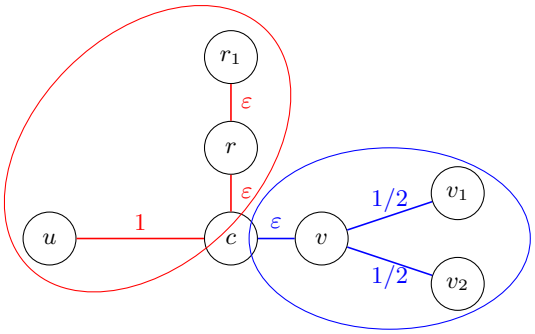


Fig. 8: Example of a tree where CMA achieves a competitive ratio of 3/2 as $\varepsilon \to 0$.

The total weight of the graph is $W = 2+3\varepsilon$, so the agents receive this budget in CMA. The centroid is at c. Agent 1 explores the child subtree T_r and does not have enough budget to explore T_v, but does have enough to explore T_u, arriving at c at exactly $2+3\varepsilon$. Agent 2 has enough budget to go to c and explore T_v, in total time also exactly $2+3\varepsilon$. This means both agents use their budgets, explore the whole graph, and meet at c at $2+3\varepsilon$. Now if the exit is at u, then the agents need one unit of time to get there giving a total evacuation time of $3+3\varepsilon$.

On the other hand to achieve the optimum, the agents start by going to v and exploring v_1 and v_2 respectively in parallel, meeting at c again at time $1+3\varepsilon$. If they have not yet found the exit, they continue together, exploring r_1 and then u. The worst case is when the exit is at u, the agents then need $2+7\varepsilon$ time to evacuate. Together with the evacuation time of CMA, this gives a competitive ratio of

$$\frac{CMA}{OPT} = \frac{3+3\varepsilon}{2+7\varepsilon} \to \frac{3}{2} \text{ as } \varepsilon \to 0.$$