Multi-Robot Tree and Graph Exploration

Peter Brass, Flavio Cabrera-Mora *Student Member, IEEE*, Andrea Gasparri *Member, IEEE* and Jizhong Xiao *Senior Member, IEEE*

***Abstract*—In this paper we present an algorithm for the exploration of an unknown graph by multiple robots, which is never worse than depth-first search with a single robot. On trees, we prove that the algorithm is optimal for two robots. For k robots, the algorithm has an optimal dependence on the size of the tree, but not on its radius. We believe that the algorithm performs well on any tree, and this is substantiated by simulations. For trees with e edges and radius r, the exploration time is less than $\frac{2e}{k} + (1 + \frac{k}{r})^{k-1}\frac{2}{k!}r^{k-1} = \frac{2e}{k} + O((k + r)^{k-1})$ (for $r > k$, $\leq \frac{2e}{k} + 2r^{k-1}$), improving a recent method with time $O(\frac{e}{\log k} + r)$ [2], and almost reaching the lower bound $\max(\frac{2e}{k}, 2r)$. The model underlying undirected graph exploration is a set of rooms connected by opaque passages; thus the algorithm is appropriate for scenarios like indoor navigation or cave exploration. In this framework, communication can be realized by bookkeeping devices being dropped by the robots at explored vertices, the states of which are read and changed by further visiting robots. Simulations have been performed in both tree and graph exploration to corroborate the mathematical results.**

***Index Terms*—Path Planning for Multiple Mobile Robot Systems, Mapping, Distributed Robot Systems**

I. Introduction

THE exploration of a completely unknown environment by mobile robots has received attention for a long time, as long as there have been mobile robots, for the first task of an autonomous robot is to find his way around. This holds whether the robot is a Mars Rover, a household cleaning appliance, or on a search-and-rescue mission in a collapsed building. The problem has been well-studied with many different models for a *single* robot exploring the environment, under line-of-sight or distance sensing constraints, in obstacle-dense or sparse environments, with various motion constraints and many other model variants. The situation is much less clear for exploration by *multiple* robots.

In this paper, we consider the situation of multiple robots exploring an obstacle-dense environment, modeled as a graph, from a single starting vertex. The graph is initially unknown; existence of edges becomes known only when a robot sees one end of the edge from a vertex, and the other end of the edge becomes known only when the robot actually follows that edge. This models an environment of sites with passages between them, where the passages are opaque: from either end it is not clear where the passage goes. All edges have unit length, and each robot can follow one edge in each time step. In particular, this work introduces a strategy that explores any tree in time $\frac{2e}{k} + O((k + r)^{k-1})$, improving a recent method with time $O(\frac{e}{\log k} + r)$ [2]. Our strategy also promises efficient exploration on general graphs.

The rest of the paper is organized as follows. In Section II the state of the art for the exploration problem is given. In Section III, the model details and the proposed algorithm are described. In Section IV a theoretical analysis concerning the algorithm performance on general graphs and on trees is proposed. In Section V, a simulation scenario and results in both tree and graph-like environments are provided and finally in Section VI conclusions are drawn and future work is discussed.

Manuscript originally submitted on December 15, 2009. This work is partially supported by the National Science Foundation of the USA under grant IIS-0644127

This paper is an improved version of our conference paper [1]

P. Brass, and J. Xiao (corresponding author) are with the Departments of Computer Science and Electrical Engineering, The City College of New York, 160 Convent Avenue, New York, NY 10031, USA peter@cs.ccny.cuny.edu, jxiao@ccny.cuny.edu

F. Cabrera-Mora is with the Department of Electrical Engineering, The Graduate Center of the City University of New York, New York, NY, 10016 USA fcabrera-mora@gc.cuny.edu

A. Gasparri is with the Department of Computer Science, University of Rome “Roma Tre”, Via Vasca Navale 79, Roma 00146, Italy gasparri@dia.uniroma3.it. Part of this work was done during A. Gasparri’s stay at the Robotics Laboratory of the City University of New York as a visiting researcher.

II. Related Work

The previous work on exploration can be roughly divided into the following classes, according to the underlying model where the environment can be:

1) a geometric structure represented as union of polygonal obstacles
2) a geometric structure represented as raster cells
3) a graph structure with uniquely identifiable vertices
4) a graph structure with anonymous vertices which need to be marked to be recognized
5) a directed graph structure

Each of these models has its motivation, and has been studied in numerous variants. The first model has been studied in [3], [4], [5]; it typically assumes that the robot knows everything within line-of-sight visibility, and is thus related to Art Gallery problems [6], but differs from watchman tours [7], [8] in that the polygons are initially unknown. This model is popular in the computational geometry community, as an example we cite [9], where a competitive algorithm for exploring the inside of a simple polygon is given, and [10], where the optimal competitive ratio is studied.

The second model is more popular in the robotics community: the environment is viewed as a grid in which some cells are open, others blocked, and still others unknown, or more complicated cell states as in evidence grids [11]. This model is more compatible with diverse types of sensing, like line-of sight, fixed radius, limited viewing angle, etc. In this model, even the exploration of a mostly empty plane might be nontrivial, solved in [12], for instance, by maintaining and following the frontier of the unexplored terrain; but in an obstacle-dense environment, that frontier might decompose in many components.

The third model is the model assumed in this paper: the environment is given as a graph, nodes corresponding to locations and edges to passages between the locations. Edges are assumed to be opaque: we know where an edge leads only when we have explored it. This is a natural model, both as an abstraction of obstacle-dense environments that we may divide into cells corresponding to the graph nodes, and as a model for state space exploration when the state transitions work in both directions. The assumption that the vertices are identifiable, and will be recognized when revisited, is reasonable in this context, and an essential model property. It has long been known that depth-first search is an efficient method to explore any graph by a single robot in this model, at most by a factor two slower than the optimum exploration strategy. A number of papers studied the influence of further information, and decreased the factor-two gap for specific graph classes [13], [14], or simulated breadth-first search, where the robot always maintains a short return path to the start vertex [15], [16], [17].

The fourth model, which differs from the third by the nodes being anonymous, and recognizable only by a marker placed on them, or by their degree or other abstract graph properties, comes from the labyrinth exploration setting. The question for the smallest capabilities, like how many "pebbles" or how many bits of memory, that allow an abstract robot to find its way out of a labyrinth, is a classic and much-studied question in the theory of computation [18], [19], [20], [21], [22]. For real robots, the question seems irrelevant, since the robot can recognize its position by other means like odometry, GPS-coordinates, or a picture of the node environment.

The fifth model, exploring a directed graph, was studied in [23]. The situation changes fundamentally from the undirected graph by the fact that you cannot go back an edge and as such depth-first search becomes impossible. This model is equivalent to exploring the state space of an unknown finite automaton; for any input, there happens some state transition, initially unknown to us. The states correspond to vertices and the transitions to directed edges, and we recognize states we have visited before. This has been proposed in [23] as model of learning: each action makes a change on the outside world. Initially, we do not know the effect of the actions, but by trying the actions and recognizing previous states we acquire knowledge about the possible actions. This model again has been studied in theoretical computing [24], [25], [26]. [27] extends the research to exploring a directed graph by multiple robots.

Of these different exploration models, only the second (grid) has received wider study in the context of multi-robot exploration. A major problem for the grid model is the fusion of the exploration maps of the individual robots. This problem does not occur with the graph model, even when starting from a continuous or a grid model. Thus, deriving a graph as a representation is a reasonable step [28]. The graph might even be made physical by dropping nodes in the explored region [29], [30]. The frontier approach is extended to multiple robots in [31]. For multi-robot undirected graph exploration, which is our underlying model, the most relevant paper is [2].

III. The Algorithm

The proposed Multi-Robot Depth First Search (MR-DFS) algorithm is a natural adaptation of Depth First Search to parallel search by multiple robots. The idea of the algorithm is simple: an edge is considered finished if a robot followed that edge, and returned by that same edge: by this we assume that he has explored everything that can be reached by that edge. As long as there are unfinished edges, the robot selects one of them to explore; only if all edges have been finished, he returns by that edge by which he originally entered the vertex.

This natural strategy can be used in many settings; most relevant to real implementation would be a completely asynchronous movement of the robots. For our analysis, we assume the robots to move synchronously in time-steps, and we want to minimize the total number of time-steps before the robots return to the start vertex and declare the search finished. In each time-step, we assume that robots standing at the same vertex have an initial negotiation phase in which they decide which robot takes which edge. The robots at the same vertex announce one after another which edge they will follow, each robot's decision being based on the edges which are already taken. Since this is wireless communication between robots standing at the same vertex, we can assume it to be instantaneous, and not contribute to the duration of exploration.

Beyond this local communication, our algorithm requires only very weak communication between the robots: a robot arriving at a vertex must be able to see whether this vertex has been visited before, and if yes, by which edges robots have left the vertex, and by which edges robots returned. This communication is classically achieved for human explorers by leaving chalk-marks on the exits; for robots, the first robot to enter a vertex could drop a bookkeeping device, e.g., an RFID, on which every robot who visits this vertex registers the sequence of his entering and leaving edges. Note that additional communication appears not useful, since in our lower bound we allow complete shared information, and the algorithm almost reaches the lower bound even with this vertex-local information only. Furthermore, this is the same communication model used in [2].

Algorithm 1 provides a description of the MR-DFS for general graphs. On trees, the algorithm becomes simpler since all robots enter a vertex by the same edge for the first time, coming from the root, and it cannot happen that a robot reenters a vertex by a different edge than that by which he left it. At each vertex, a bookkeeping device is dropped by the first robot to visit that vertex, and updated by all further robots on every visit. MR-DFS requires the following minimal set of information to be stored at each vertex:

- the number of edges converging in this vertex.
- the ID of the robots that have visited this vertex before.
- for each of these robots, the original entrance edge of the robot

Algorithm 1 Algorithm Multi-robot DFS - general graph version

1: Let rob_i be a robot arriving at a vertex v through edge e
2: **if** rob_i has been at v before, and the edge e by which he returned is different from the edge by which he last time left v **then**
3: Mark e as finished edge, go back through edge e.
4: **else**
5: Either v is a new vertex for rob_i, or he returned to v after exploring the component to which edge e leads.
6: **if** rob_i has never been at v before **then**
7: Mark e as the original entrance edge of rob_i to v.
8: **else**
9: rob_i has been at v before, and returned by the same edge e by which last time he left v
10: Mark e as finished edge.
11: **end if**
12: **if** there is an edge leaving v that is neither finished, nor the original entrance edge of any robot to v **then**
13: Choose one of those edges, preferring edges that have been used by the least number of other robots before, and leave v by that edge.
14: **else**
15: Return from v by rob_i's original entrance edge.
16: **end if**
17: **end if**

Algorithm 2 Algorithm Multi-robot DFS - tree version

Let rob_i be a robot arriving at a vertex v through edge e
Either v is a new vertex for rob_i, or he returned to v after exploring the subtree to which edge e leads.
if v is a new vertex, not visited by any robot before **then**
Mark e as the original entrance edge to v.
end if
if rob_i has been at v before **then**
Mark e as finished edge.
end if
if there is an edge leaving v that is neither finished, nor the original entrance edge to v **then**
Choose one of those edges, preferring edges that have been used by the least number of other robots before, and leave v by that edge.
else
Return from v by the original entrance edge.
end if

- For each edge, the IDs of the robots entering and leaving through that edge

Thus, every edge that is followed by a robot will be recorded, including the direction, by the bookkeeping devices at either end. In a real implementation, one has to consider the very limited storage capacity of RFID tags, and use it most efficiently. If we assume that each robot entering a vertex will find the same exits, and find these in the same sequence (e.g., starting north and enumerating clockwise), we need to store for each exit only if a robot has entered the vertex through that exit (so it is either finished or original entry edge), or, if not, the number of robots that have left through that edge. This information is sufficient for the algorithm and its analysis; the actual identity of the robots need not be stored on the bookkeeping device.

To summarize, each robot running the MR-DFS algorithm follows essentially a tree, starting at the common start vertex. If he meets his own tree by a different edge, he immediately leaves along that edge again (line 2–3, Algorithm 1). If he meets another robot's tree, they divide the outgoing edges for exploration, each choosing some unexplored edges, as long as possible (line 12–13: Algorithm 1). At any time and for each robot that has visited a vertex, there is at most one edge by which that robot left without returning back. If several robots jointly explore the outgoing edges of a vertex, and a returning robot finds no unexplored edge any more, he will join another robot in the branch the other is just exploring. Only if each edge has been followed by a robot in both directions, the robot returns from that vertex by his original entrance edge (line 7: Algorithm 1).

Fig. 1 shows two robots exploring a graph from a common starting vertex, with their path after 5, 8, 11, and 15 steps. The dotted line represents the path of robot rob_a, and the dashed line represents the path of robot rob_b.

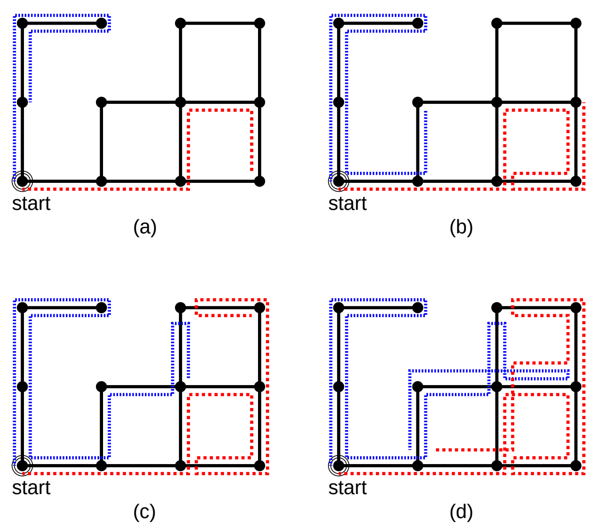


Fig. 1: Path of two robots after a) 5, b) 8, c) 11, and d) 15 steps

IV. Theoretical Analysis

In this section, a theoretical analysis of the MR-DFS algorithm is proposed. The goal is to provide a characterization of the MR-DFS exploration time on general graphs and trees.

A. Preliminaries

Let us consider a graph $\mathcal{G} = \{V, E\}$ modeling an environment to be explored. The graph is considered to be completely explored only if every edge is followed by at least one robot and all the robots return to the starting vertex. This requirement that the robots return to the starting point at most doubles the exploration time, since they could just follow their way back. The number of rounds required in our model to completely explore the graph is the exploration time t_c.

If there is only one robot, the exploration time for a graph is at least $e = |E|$, since every edge needs to be followed. If the underlying graph is a tree, then every edge the robot follows outward he must also use coming back, so the exploration time is at least $2e$. Classical depth-first search (DFS) does explore any graph with one robot in $2e$ steps. Thus, the single-robot scenario has an easy solution, which is optimal for trees and at most a factor two slower for arbitrary graphs.

Notice that if we aimed to optimize the total number of steps taken by all robots together, instead of the number of rounds, then the availability of multiple robots would not help: they still need to follow $2e$ edges to explore a tree, and we can do that with a single robot using DFS, so for that measure, parking all but one robot at the start vertex and using DFS for that last robot would be an optimal solution.

If there are k robots available, the best we can hope for is a speed-up of a factor k. In each round, k new edges are explored, so we need at least $\frac{e}{k}$ rounds for a general graph, and $\frac{2e}{k}$ rounds for a tree. This speed-up is not always possible. If the graph is just one long path of length r from the starting vertex, one robot would need to travel all the length r and return back, regardless of the number of robots there might be available at the common starting vertex. If r is the radius of the graph, i.e., the longest distance from the starting vertex to any other vertex, then one of the robots has to reach that vertex at maximum distance, and come back. So we have two lower bounds for the exploration time t_c.

- e/k, since each edge needs to be visited by a robot;
- $2r$, since a vertex at maximum distance must be visited.

Therefore, for a given graph $\mathcal{G}$ with e edges and radius r, the *lower bound* for the exploration time is $\max(e/k, 2r)$, and $\max(2e/k, 2r)$ if it is a tree. Since the optimum strategy, which knows the graph in advance and just has to visit all edges, takes at least this time, then any algorithm that is within some factor of that lower bound is competitive and of interest.

B. Analysis on general graphs

In order to characterize the exploration time of the MR-DFS on general graphs two important properties must be introduced.

***Lemma** 1:* In the MR-DFS algorithm, each edge is used by each robot at most once in either direction.

Proof: To see the claim of this lemma, we assume that robot rob_i follows the edge uv from u to v twice, at time t_1 and t_2. Between these times, rob_i returns at least once to u. Each time rob_i returns by a different edge than vu, he will immediately go back by the edge by which he came (line 2-3, algorithm 1). So rob_i must return once by vu, but then he marks uv as finished and will not follow this edge a second time. ■

***Lemma** 2:* In the MR-DFS algorithm, all robots finish their exploration at the same time step.

Proof: To prove this lemma, suppose that a robot rob_1 has already returned to the origin and found no further eligible edge, declaring the search finished, whereas rob_2 is still out at a different vertex at that same time step. The robot rob_2 is connected to the start vertex by his return path $v_p, v_{p-1}, \ldots, v_1$, with v_p being the current position of rob_2, v_1 the start vertex, and $v_{q-1}v_q$ being the original entry edge of rob_2 to v_q for $q = 2, \ldots, p$.

The edge v_1v_2 was not eligible for rob_1, otherwise rob_1 would have followed that edge. There are two possible reasons why an edge can become ineligible; either it is finished, with a robot going and returning by that edge, or it is the original entry edge of a robot to that vertex. No edge can be the original entry edge in both directions, since it becomes ineligible in the opposite direction as soon as it is first used. Since the edges along the path $v_1, v_2, \ldots, v_p$ are original entry edges of the robot r_2, they cannot be original entry edges in the opposite direction. Thus every edge along this path is either finished or eligible. Let $v_{i-1}v_i$ be the last edge on the path $v_1, \ldots, v_p$ that is finished, and let rob_3 be the robot that finished this edge. Consider the time step when rob_3 finished this edge. Since rob_2 used the edge before it was finished, the edge is somewhere on the return path of rob_2 at that time. If rob_2 is not at the same vertex as rob_3, then there is an eligible edge on the return path of rob_2 from v_i in the direction of rob_2. Thus, rob_3 would have followed that edge instead of returning by v_iv_{i-1}. As such, rob_2 and rob_3 must be at the same vertex at that time step, they both do not find any eligible edge, and return together.

The same argument applies to any previous edge along that path. At the time immediately before rob_2 and rob_3 return together, the edge $v_{i-1}v_i$ was still eligible. But then none of the earlier edges along that path can be finished, since for each vertex there is still one eligible edge available. Consequently, rob_1 at the start vertex has still an eligible edge available, giving a contradiction to our initial assumption. ■

Let us now state the main result concerning the exploration time of the MR-DFS algorithm on general graphs.

***Theorem** 1:* The algorithm MR-DFS explores any connected graph with e edges, traversing each edge, in at most $2e$ steps.

Proof: The proof of the theorem is a consequence of the previous lemmas. In particular, according to lemma 1 a robot uses each edge at most once in each direction. Therefore, in the worst case scenario, all the robots are going to traverse $2e$ edges. Furthermore according to lemma 2 all the robots finish the exploration at the same time. At this point, since at each step only one edge can be traversed, the number of edges that each robot can traverse is at most $2e$. ■

***Remark** 1:* An important consequence of Theorem 1 is that the MR-DFS algorithm explores any graph *completely*, and is *never worse* than classical single-robot DFS.

C. Analysis on trees

The proposed MR-DFS algorithm is on trees generally much better than a single-robot DFS.

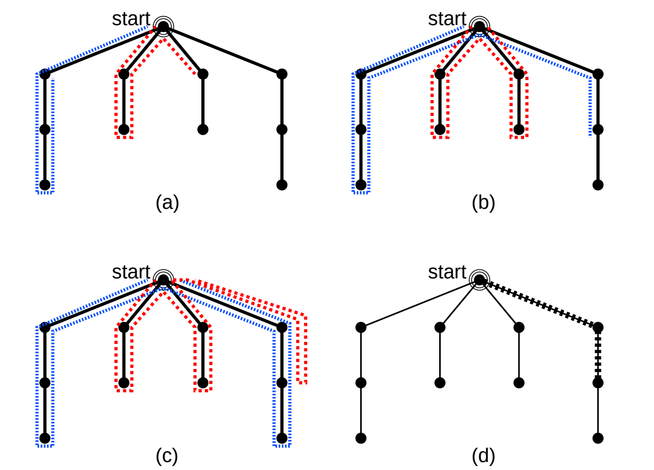


Fig. 2: Path of two robots after a) 5, b) 8, and c) 12 steps on a tree of degree 4. d) Shows the edges traversed by both robots

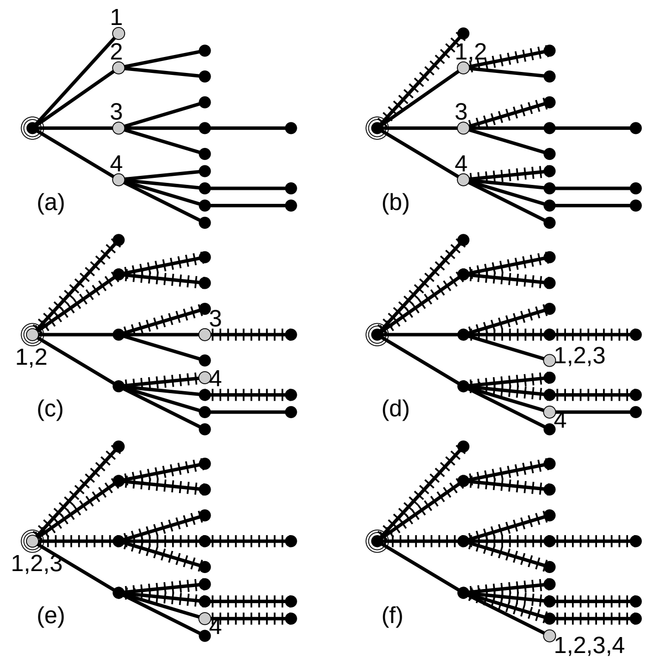


Fig. 3: Decreasing branching property: worst-case scenario. Path of the robots after a) 1 step, b) 3 steps, c) 6 steps, d) 8 steps, e) 10 steps, and f) 12 steps

Fig. 2 shows two robots exploring a tree of degree 4, showing the state after 5, 8, and 12 steps, and the edges used by both robots. Again, the dotted line represents the path of robot rob_1, and the dashed line the path of robot rob_2. At the beginning, each robot enters a branch that has not been used before. Only the last branch is entered by both robots. The robot that entered the last branch second (rob_2) meets after two steps the returning robot (rob_1), that entered the branch first, and they both return together to the starting vertex.

The fundamental property of the MR-DFS algorithm on trees is the decreasing branching property as described in the following lemma. To this end, let us first define an incoming edge of a vertex as the edge in the direction of the starting vertex, and all other edges as outgoing edges.

***Lemma** 3:* The edges used by several robots form a subtree. If a vertex is visited by j robots, then among the outgoing edges there is at most one edge that is taken by all j robots, and at most $i+1$ edges that are taken by at least $j-i$ robots, for $i = 0, \ldots, j-1$.

Proof: To prove this lemma, we consider a vertex v that has d outgoing edges and is entered by j robots. Fig. 3 illustrates the worst-case situation of the lemma for $d = j = 4$. Each robot that enters this vertex chooses an outgoing edge, explores a subtree, returns to the vertex and chooses another edge, and so on, until it finds no further edges left. Each time it returns from an edge, that edge becomes finished and unavailable for all robots which have not already used it. We number the outgoing edges $e_1, \ldots, e_d$ in the sequence in which robots return from that edge. Thus the first robot to return to v blocks e_1 for all those robots which have not already entered it. Since other robots can have entered e_1 only after all other edges had been entered by at least one robot, then

- if $d \geq j$, no other robot can have entered e_1, so e_1 is used by only one robot;
- else $j > d$, so at most $j - d$ other robots entered e_1, and e_1 is used by at most $j - d + 1$ robots.

In the same way, for $1 \leq a \leq d$ the edge e_a is blocked for all robots that have not entered it at the time the robot on it returns. Any further robot can have entered this edge only after all $d-1$ other edges have been entered by at least one robot; available for that are the $j-1$ other robots, which are available $a-1$ additional times from their previous returns. Thus

- if $d-1 \geq j+a-2$, no other robot can have entered e_a, so e_a is used by only one robot;
- else $d-1 < j+a-2$, then at most $(j+a-2)-(d-1)$ other robots entered e_a, and e_a is used by at most $j+a-d$ robots.

So if $j \geq d$, then the d outgoing edges are used by at most $j, j-1, \ldots, j-d+1$ robots. If $j < d$, then the outgoing edges are used by at most $j, j-1, \ldots, 1, 1, \ldots, 1, 1, 1$ robots. This completes the proof of the Lemma. ■

Let us now introduce the concept of *excess multiplicity* $\mu(e_i)$ of an edge e_i as the number of additional robots after the first that use that edge. By Lemma 1, each robot uses each edge at most twice, going out and returning, so for each edge e_i we have $0 \leq \mu(e_i) \leq k-1$, and the edge is used exactly $2 + 2\,\mu(e_i)$ times. Fig. 4 shows the multiplicity of a subtree with 3 outgoing edges being explored by 4 robots. The excess multiplicity plays a key role to define an upper

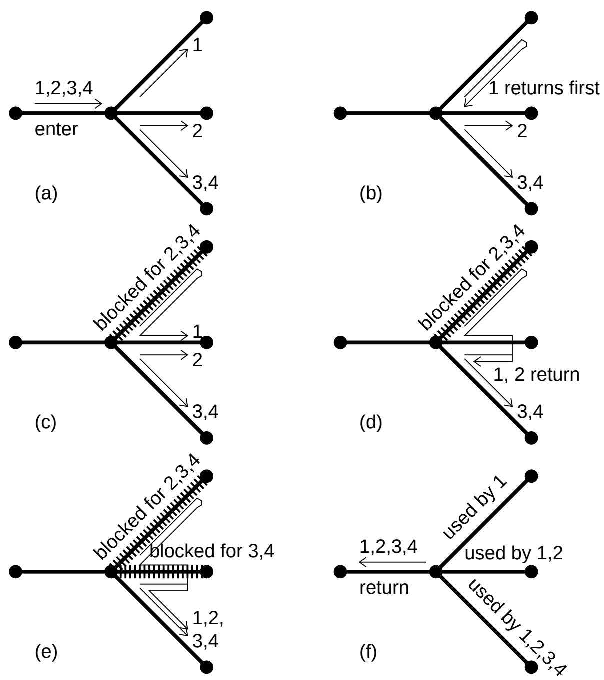


Fig. 4: Multiplicity of a subtree with 3 edges being explored by 4 robots

bound for the exploration time of the MR-DFS algorithm as described by the following lemma.

***Lemma** 4:* The time that the MR-DFS algorithm takes to explore a tree with e edges by k robots is

$$t_c = \frac{1}{k}\left(2e + 2\sum_{e_i}\mu(e_i)\right). \tag{1}$$

Proof: To obtain the bound on the total exploration time, we just add up the work done by each robot, and divide by k: since all the robots finish at the same time, we just count the total number of edges walked by the robots when they finish. Each edge was taken at least once in each direction, plus $2\sum_{e_i}\mu(e_i)$ additional edges, taken by several robots (multiplicity). ■

***Lemma** 5:* The function $f(k,r)$ defined by $f(k,1) = \binom{k}{2}$ and the recursion

$$f(k,r) = \binom{k}{2} + \sum_{i=2}^{k} f(i, r-1) \tag{2}$$

is

$$f(k,r) = \binom{k+r}{k-1} - k. \tag{3}$$

Proof: We solve the recursion by repeated application of the binomial sum

$$\binom{a}{a} + \binom{a+1}{a} + \cdots + \binom{b}{a} = \binom{b+1}{a+1} \tag{4}$$

We have

$$\begin{aligned} f(k,2) &= \binom{k}{2} + \sum_{i=2}^{k} f(i,1) \\ &= \binom{k}{2} + \sum_{i=2}^{k}\binom{i}{2} = \binom{k}{2} + \binom{k+1}{3}. \end{aligned}$$

We apply this again and find

$$\begin{aligned} f(k,3) &= \binom{k}{2} + \sum_{i=2}^{k} f(i,2) \\ &= \binom{k}{2} + \sum_{i=2}^{k}\left(\binom{i}{2} + \binom{i+1}{3}\right) \\ &= \binom{k}{2} + \sum_{i=2}^{k}\binom{i}{2} + \sum_{i=3}^{k+1}\binom{i}{3} \\ &= \binom{k}{2} + \binom{k+1}{3} + \binom{k+2}{4}. \end{aligned}$$

From this, we prove

$$f(k,r) = \binom{k}{2} + \binom{k+1}{3} + \cdots + \binom{k+r-1}{r+1} \tag{5}$$

by induction, using

$$\begin{aligned} f(k,r) &= \binom{k}{2} + \sum_{i=2}^{k} f(i, r-1) \\ &= \binom{k}{2} + \sum_{i=2}^{k}\sum_{j=0}^{r-2}\binom{i+j}{2+j} \\ &= \binom{k}{2} + \sum_{j=0}^{r-2}\sum_{i=2}^{k}\binom{i+j}{2+j} \\ &= \binom{k}{2} + \sum_{j=0}^{r-2}\binom{k+1+j}{3+j} \\ &= \sum_{j=0}^{r-1}\binom{k+j}{2+j}. \end{aligned}$$

Finally, we reduce this sum by

$$\begin{aligned} f(k,r) &= \binom{k}{2} + \binom{k+1}{3} + \cdots + \binom{k+r-1}{r+1} \\ &= \binom{k}{k-2} + \binom{k+1}{k-2} + \cdots + \binom{k+r-1}{k-2} \\ &= \binom{k+r}{k-1} - \binom{k-1}{k-2} - \binom{k-2}{k-2} \\ &= \binom{k+r}{k-1} - (k-1) - 1. \end{aligned}$$

■

Let us now state the main result concerning the exploration time of the MR-DFS algorithm on trees.

***Theorem** 2:* The MR-DFS algorithm explores a tree with e edges and radius r using k robots in time is at most

$$\min\left(2e, \frac{2e}{k} + \frac{2}{k}\binom{k+r}{k-1}\right) < \frac{2e}{k} + \left(1 + \frac{k}{r}\right)^{k-1} \frac{2}{k!} r^{k-1}. \tag{6}$$

Proof: The proof comes from the observation that the maximum total *excess multiplicity* of all multiply used edges together is the sum of the excess multiplicities of the subtrees entered from the root, plus the excess multiplicities on the edges from the root to those subtrees. In a tree of radius r, each subtree entered from the root has radius at most $r-1$, and by Lemma 3 there is at most one subtree entered by all k robots, at most two subtrees entered by $k-1$ or k robots, etc., and only at most $k-1$ subtrees are entered by two or more robots. All other subtrees entered from the root are entered only by one robot, so they do not contribute any excess multiplicity. Thus the maximum total excess multiplicity $g(k,r)$ as a function of the number of robots k and the radius r, satisfies the recursion

$$\begin{aligned} g(k,r) \quad \leq \quad & \binom{k}{2} + g(k,r-1) + g(k-1,r-1) + \\ & g(k-2,r-1) + \cdots + g(2,r-1) \end{aligned}$$

with the boundary condition $g(k,1) = (k-1) + (k-2) + \ldots + 1 = \binom{k}{2}$. This is the same recursion as the one solved in Lemma 5, only with $\leq$ instead of $=$; so $g(k,r) \leq f(k,r)$. Thus the total excess multiplicity is bounded by $g(k,r) \leq \binom{k+r}{k-1} - k$. From Lemma 4 this bounds the exploration time as

$$t_c = \frac{1}{k}\left(2e + 2f(k,r)\right) < \frac{2e}{k} + \frac{2}{k}\binom{k+r}{k-1}. \tag{7}$$

To show the growth rate of this expression for k small and r large, we observe

$$\begin{aligned} \frac{1}{k}\binom{k+r}{k-1} \quad &= \quad \frac{(k+r)(k-1+r)\cdots(2+r)}{k(k-1)!} \\ &< \quad \frac{(k+r)^{k-1}}{k!} = \left(1 + \frac{k}{r}\right)^{k-1} \frac{1}{k!} r^{k-1}. \end{aligned}$$

For $r \geq k$ this is less than $\frac{2^{k-1}}{k!} r^{k-1} \leq r^{k-1}$; indeed, the coefficient of r^{k-1} is rapidly decreasing for larger k and $r \geq k$. ■

***Remark** 2:* In its dependence on e, this is optimal and improves the $O(\frac{e}{\log k} + r)$-algorithm of [2]. The dependence on r, however, is not. This is an interesting bound for trees with many edges and small radius (trees with high branching factors). The bound on the total *excess multiplicity* used in the proof above views it only as a function of r and k, and leaves e open. This bound can be reached, but only if e is very large compared to r. To obtain a further improvement along these lines in the bound would require an analysis with e as third parameter. Also, the bound in Lemma 3 can be improved if the number of robots is larger than the degree: if a robots reach a vertex with two outgoing edges before the first of these robot returns to that vertex, they will have distributed equally over the two edges until one of the two edges is finished by the first returning robot. At that time, $\frac{1}{2}a$ robots will have entered each branch, so the total multiplicities of the edges are at most a and $\frac{1}{2}a$, which is much better than a and $a-1$ for large a. For small number of robots, no improvement can be expected, as the next theorem shows.

For two robots ($k=2$) the following Theorem 2 shows a type of optimality of MR-DFS: no strategy can guarantee a better competitive ratio against an optimal explorer, who already knows the tree and always makes the best choices.

***Theorem** 3:* The MR-DFS algorithm with two robots explores a tree with e edges and radius r in time at most $e+r$. This is at most $\frac{3}{2}$ of the optimum exploration time, and no algorithm for two robots guarantees a factor less that $\frac{3}{2}$.

Proof: For two robots ($k=2$) Lemma 3 implies that the subtree used by both robots does not branch, so it is a path with length at most r. Thus, $\sum_i \mu(e_i) \leq r$. By Lemma 4, the exploration time is at most $\frac{1}{2}(2e+2r) = e+r$, as claimed by the theorem. Furthermore, as explained in Section IV-A, the general lower bound of the exploration time of a tree using two robots is $\max(\frac{2e}{2}, 2r) = \max(e, 2r)$, and $e + r \leq \frac{3}{2}\max(e, 2r)$:

- for $r \leq \frac{1}{2}e$ we have that $\max(e,2r) = e$ and $e+r \leq \frac{3}{2}e$,
- for $r \geq \frac{1}{2}e$ we have $\max(e,2r) = 2r$ and $e + r \leq 3r = \frac{3}{2}2r$.

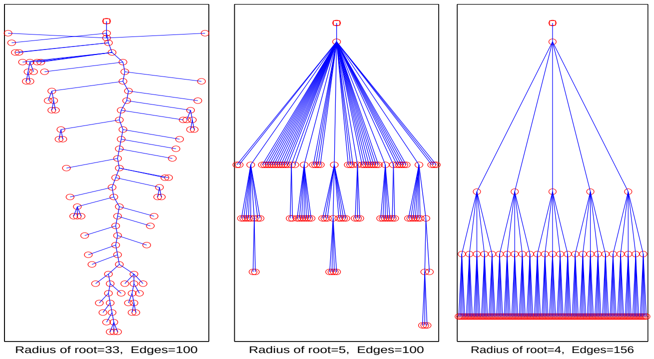
This proves the upper bounds of the theorem.

To see that no algorithm can guarantee a better approximation ratio than $\frac{3}{2}$ for the optimum exploration time, we use an adversarial construction. Consider a graph which has three branches, two of length t and one of length $2t$. This tree can be optimally explored by two robots in time $4t$: one robot explores the two short branches, the other the one long branch. But any algorithm finds out whether a branch is a short branch or a long branch only after a robot has reached the end of the branch. Thus an adversary who reveals the graph as it is explored can always make the last branch to be explored a long branch; so any algorithm can be forced to take exploration time at least $6t$. Thus no algorithm for two robots gives a better competitiveness ratio than the $\frac{3}{2}$ achieved by the MR-DFS algorithm. ■

***Remark** 3:* The adversarial construction described above is the special case of a general construction described in [32], [2], which shows that with k robots, no strategy can guarantee a competitive factor better than $2 - \frac{1}{k}$.

V. Simulation Results

Two set of simulations were performed in order to corroborate the most important results of this paper: Theorem 2 and Theorem 3. To do so, the algorithm was run in three different scenarios: *long*, *wide* and *full N-ary trees* (from now on we will refer to *full N-ary trees* as simply "*N-ary trees*"). Examples of these scenarios are shown in figure 5. The size of the tree was increased (in long and wide trees) by increasing the number of



(a) Long tree (b) Wide tree (c) N-ary tree (N=5)

Fig. 5: Example of different tree generation methods used for the simulations.

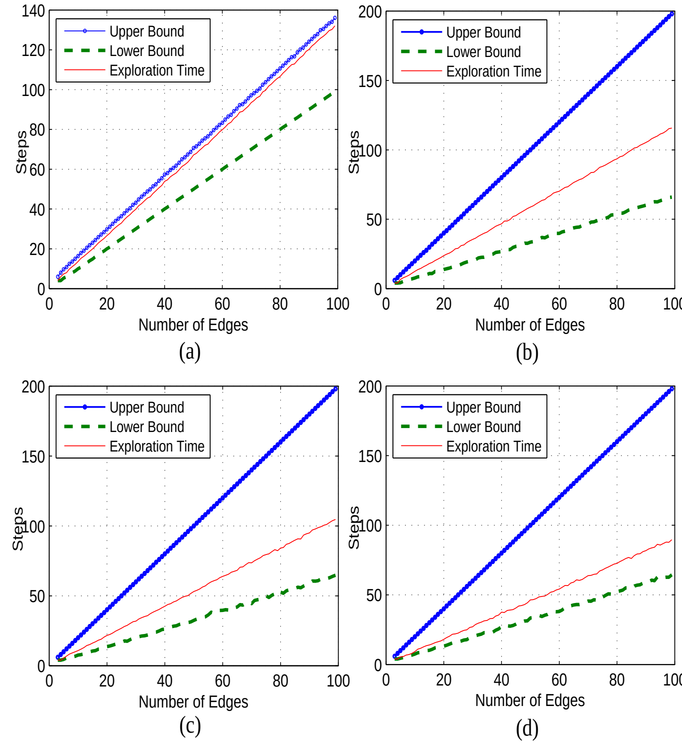


Fig. 6: Comparison of exploration times and bounds of exploration on long trees of increasing number of vertices using a) 2, b) 3, c) 5, and d) 20 robots.

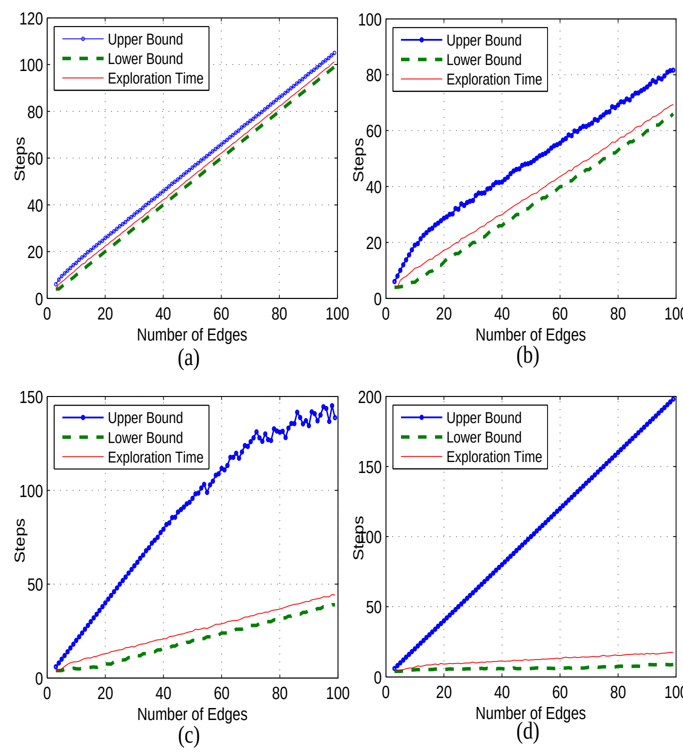


Fig. 7: Comparison of exploration times and bounds of exploration on wide trees of increasing number of vertices using a) 2, b) 3, c) 5, and d) 20 robots.

edges. In N-ary trees, the size of the tree was defined by the number of children N that each vertex was allowed to have and by the radius of the tree.

The way the robots were distributed in a vertex is as follows: assume k robots arrive at a vertex v that has e_v downward unexplored edges. The k robots will distribute themselves in the most homogeneous way possible where the maximum difference in the number of robots in every edge is equal to one. As an example consider 5 robots arriving at a vertex with 3 unexplored downward edges: two of those edges will be taken by 2 robots and the last edge will be taken by only one robot. The idea is to obtain the maximum parallelism in the exploration process. For long and wide trees, since the same number of edges *e* can produce very different configurations of trees (each one with a different exploration time), we performed 100 runs of the simulation per each value of *e*. The results of the first set of simulations are shown in Fig. 6 and Fig. 7. The plots show the upper bound defined by Theorem 2, the lower bound ($\max(2e/k, 2r)$) and the exploration time (mean of 100 runs) due to different numbers of robots exploring the tree.

For N-ary trees, only one simulation per tree configuration was run since, due to the symmetry of the tree, the algorithm will make the robots explore the tree in the same way all the time. The plots in Fig. 8 show the upper bound (straight line) and exploration time (dashed line) for this type of tree when the exploration is performed by different number of robots from 2 to 6. The results for lower bound were not shown in order to simplify the reading of the plots.

From the results of this set of simulations (Fig. 6 to Fig. 8) we can observe that the bounds of exploration, as defined in this paper, hold at all times. An interesting result is shown in Fig. 6 and Fig. 7 when using 2 robots: the curve of the upper bound of Theorem 2 matches tightly that of the actual exploration time of the algorithm. In particular in wide trees, the analysis presented in section IV produces bounds of exploration that perfectly enclose the exploration time. Recall that tightness on the bounds of exploration is desired in order to perform estimations on the actual exploration time when no explicit expression for this exploration time has been found (like in this case).

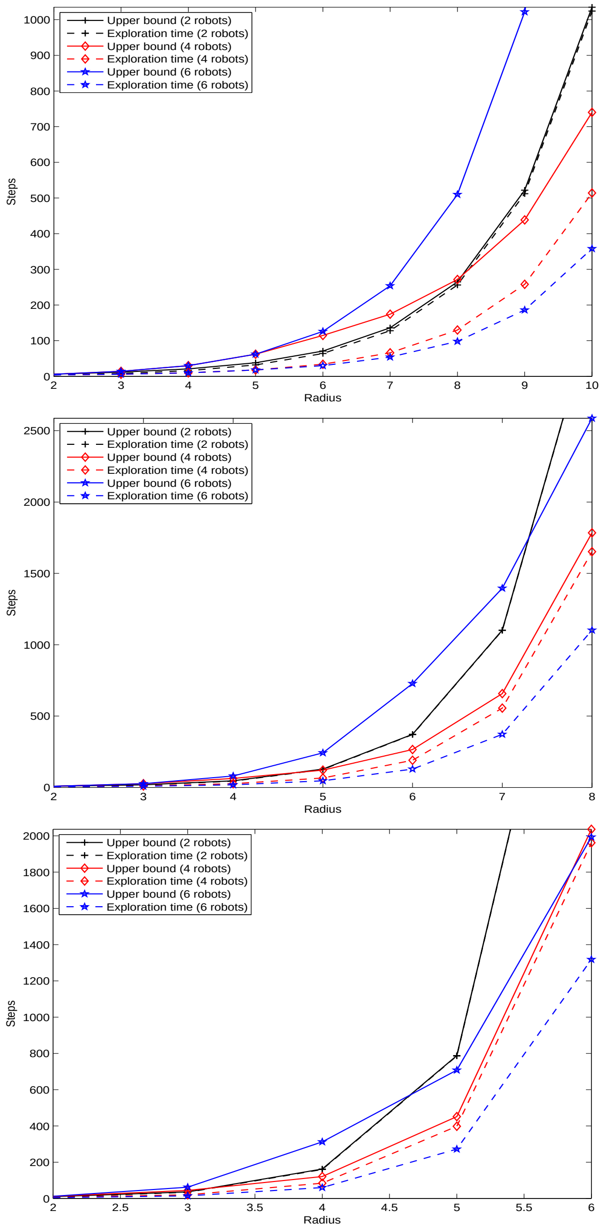


Fig. 8: Comparison between upper bound (straight line) and exploration time (dashed line) on N-ary trees of increasing radius and different number of robots. Subfigures from top to bottom respectively show the curves for N=2, N=3 and N=5.

From the results on wide trees (Fig. 7) it is evident that our lower bound is very close to the exploration time. As such, it suggest the existence of a linear function of k, r and e that actually defines the exploration time or that at least upper-bounds it more tightly. The results on all trees corroborate Remark 2, since our upper bound is indeed prevalent on trees with many edges and small radius (wide trees), particularly when using a small number of robots. For long trees, the upper bound is basically defined by $2e$.

The results on all trees also show that our MR-DFS algorithm is effective in reducing the exploration time when increasing the number of robots and that this exploration time is at all times better than the single robot depth first search approach (a desired characteristic of any multi-robot strategy). Lastly, we can observe from the simulations that when increasing the number of robots, the exploration time of the algorithm is brought down closer to the lower bound. That is, the exploration time is reduced closer to the optimal time of exploration.

Fig. 9 shows the behavior of the algorithm on a tree with a fixed configuration when using up to 15 robots. The fixed configuration corresponds to a N-ary tree (N=7) and a radius of 5. The plot clearly shows how the exploration time is consistently reduced when more robots are included in the system.

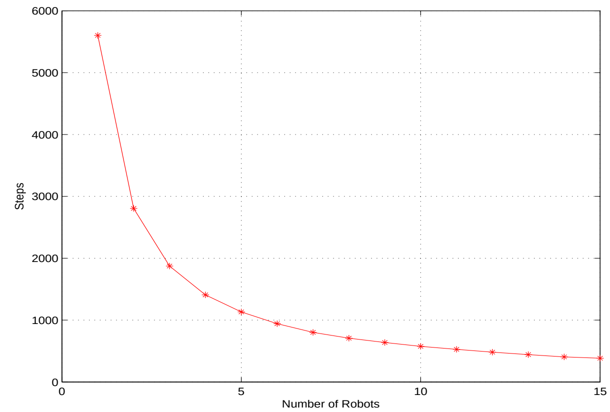


Fig. 9: Exploration time for increasing number of robots in a tree with a fixed configuration (N-ary tree with N=7 and r=5).

A second set of simulations was performed to corroborate the statement of Theorem 3: with 2 robots the upper bound of the exploration time is $e + r$. Fig. 10 shows the results for long and wide trees.

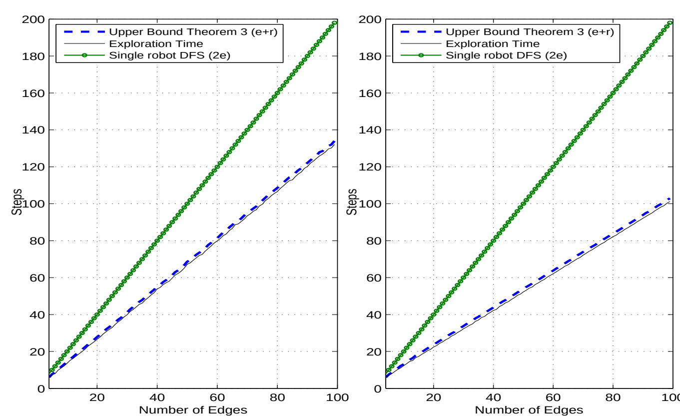


Fig. 10: Comparison between the exploration time and the upper bound defined in Theorem 3 on long (left) and wide (right) trees of increasing number of edges using 2 robots.

The plots show that the upper bound holds at all times and that, as expected, it is tight with respect to the actual exploration time. Fig. 10 also allows us to observe in detail the performance of the algorithm using two robots and how it contrasts with the result of single robot DFS: in wide trees the average reduction in the exploration time is approximately 50%, whereas in long trees the reduction averages 30%.

VI. CONCLUSIONS

In this paper, we have proposed an algorithm multi-robot DFS for the exploration of an unknown undirected graph, which is guaranteed to succeed on any graph, never worse than classical single-robot DFS, and which on trees we have proved to be optimal for two robots, and having optimal dependence on the size of the tree, but not its radius, for k robots. In this specific graph exploration scenario, the robots are initially all located at a common starting vertex, they discover the existence of an edge only when they see one end of it, and know where an edge leads only when they have followed it. Vertices that have been visited before are recognized.

The proposed algorithm needs only a local communication model, where communication happens only between a robot, and a bookkeeping device left at that node, or between robots standing simultaneously at the same node. So the robots are almost completely unaware of the actions of the other robots. The bookkeeping devices are not in contact with each other; they could be replaced by a piece of chalk leaving marks on the possible exits of the rooms. This is a much weaker communication assumption than global shared information; if global shared information is available, no bookkeeping devices are needed. The exploration algorithm will even succeed if some robots are lost or destroyed. As long as there are edges that are not marked as finished, some other robot will follow up that edge. If there is at least one robot left, only an incorrect finished mark can keep a vertex from being visited. Destroying or manipulating the marks on the bookkeeping devices can prevent the exploration from success: erasure of finished marks can keep the algorithm from terminating.

In addition to our theoretic analysis, several simulations have been performed in order to corroborate the mathematical results previously described. The result of the simulations show that our analysis on trees produces upper and lower bounds on the exploration time that are close to the actual exploration time of the algorithm, particularly when considering two robots. The simulations also show that the algorithm effectively reduces the exploration time when the number of robots is increased and that this exploration time is at all times better than when using the single-robot depth first search approach. Moreover, it can be seen how the performance of the algorithm reaches closer to the optimal exploration time when more robots are used to perform the exploration.

The analysis of this algorithm was only for trees; the most important next theoretical problem is to provide an analysis for general graphs. No bounds on multi-robot exploration of general graphs in this scenario are known. The bound for trees could be improved, perhaps even giving optimality for further small numbers of robots. And the most important problem for the practical applicability of this algorithm is to remove the assumption of robot movement in time-steps: the real robot movement is asynchronous, and the algorithm itself makes no assumption on synchronization; that is artifact of the analysis.

REFERENCES

[1] P. Brass, A. Gasparri, F. Cabrera-Mora, and J. Xiao, “Multi-robot tree and graph exploration,” in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2332–2337.

[2] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc, “Collective tree exploration,” *Networks*, vol. 48, pp. 166–177, 2006.

[3] C. Papadimitriou and M. Yanakakis, “Shortest paths without a map,” *Theoretical Computer Science*, vol. 84, pp. 127–150, 1991.

[4] X. Deng, T. Kameda, and C. Papadimitriou, “How to learn an unknown environment,” in *Proceedings of the IEEE 32nd Annual Symposium on Foundations of Computer Science (FOCS)*, 1991, pp. 298–303.

[5] A. Blum, P. Raghavan, and B. Schieber, “Navigation in unfamiliar geometric terrain,” in *Proceedings of the twenty-third annual ACM symposium on Theory of computin (STOC)*, 1991, pp. 494–504.

[6] J. O’Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[7] W. Chin and S. Ntafos, “Optimum watchman routes,” in *Proceedings of the twenty-third annual ACM Symposium on Computational Geometry (SoCG)*, 1986, pp. 24–33.

[8] S. Ntafos and L. Gewali, “External watchman routes,” *The Visual Computer*, vol. 10, pp. 474–483, 1994.

[9] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, “The polygon exploration problem,” *SIAM Journal on Computing*, vol. 31, pp. 577–600, 2001.

[10] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen, “Competitive online approximation of the optimal search ratio,” *SIAM J. Computing*, vol. 38, pp. 881–898, 2008.

[11] H. Moravec, “Sensor fusion in certainty grids for mobile robots,” *AI Magazine*, vol. 9, pp. 61–74, 1988.

[12] B. Yamauchi, “Frontier-based approach for autonomous exploration,” in *IEEE International Symposium on Computational Intelligence, Robotics and Automation*, 1997, pp. 146–151.

[13] P. Panaite and A. Pelc, “Impact of topographic information on graph exploration efficiency,” *Networks*, vol. 36, pp. 96–103, 2000.

[14] A. Dessmark and A. Pelc, “Optimal graph exploration without good maps,” in *European Symposium on Algorithms (ESA)*, ser. LNCS, vol. 2461. Springer, 2002, pp. 374–386.

[15] B. Awerbuch, M. Betke, R. Rivest, and M. Singh, “Piecemeal graph exploration by a mobile robot,” in *Proceedings of the 8th Annual ACM Conference on Computational Learning Theory (COLT)*, 1995, pp. 321–328.

[16] B. Awerbuch and S. Kobourov, “Polylogarithmic-overhead piecemeal graph exploration,” in *In Proceedings of the 11th Annual ACM Conference on Computational Learning Theory (COLT)*, 1998, pp. 280–286.

[17] C. Duncan, S. Kobourov, and V. Kumar, “Optimal constrained graph exploration,” *ACM Transactions on Algorithms*, vol. 2, pp. 380–402, 2006.

[18] L. Budach, “On the solution of the labyrinth problem for finite automata,” *Elektronische Informations-Verarbeitung und Kybernetik EIK*, vol. 11, pp. 661–672, 1975.

[19] F. Hoffmann, “One pebble does not suffice to search plane labyrinths,” in *Fundamentals of Computation Theory*, ser. LNCS, vol. 117. Springer, 1981, pp. 433–444.

[20] M. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, “The power of a pebble: exploring and mapping directed graphs,” in *Proceedings of the thirtieth annual ACM symposium on Theory of Computing (STOC)*, 1998, pp. 269–278.

[21] H.Wang, M. Jenkin, and P. Dymond, “Enhancing exploration in graph-like worlds,” in *Canadian Conference on Computer and Robot Vision*. IEEE, 2008, pp. 53–60.

[22] L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang, “Tree exploration with logarithmic memory,” in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 585–594.

[23] X. Deng and C. Papadimitriou, “Exploring an unknown graph,” in *Proceedings of the IEEE 33th Annual Symposium on Foundations of Computer Science (FOCS)*, 1990, pp. 355–361.

[24] S. Kwek, “A simple depth-first search strategy for exploring unknown graphs,” in *Workshop on Algorithms and Data Structures (WADS)*, ser. LNCS, vol. 1272. Springer, 1997, pp. 345–353.

[25] S. Albers and M. Henzinger, “Exploring unknown environments,” *SIAM Journal on Computing*, vol. 29, pp. 1164–1188, 2000.

[26] R. Fleischer and G. Trippen, “Exploring an unknown graph efficiently,” in *European Symposium on Algorithms (ESA)*, ser. LNCS, vol. 3669. Springer, 2005, pp. 11–22.

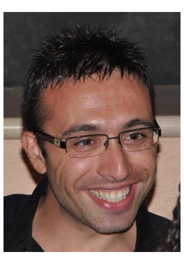
[27] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro, “Map construction of unknown graphs by multiple agents,” *Theoretical Computer Science*, vol. 385, pp. 37–48, 2007.

[28] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “Robotic exploration as graph construction,” *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 859–865, 1991.

[29] M. Batalin and G. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal*, vol. 26, pp. 181–196, 2004.
[30] ——, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, pp. 661–675, 2007.
[31] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Sec. Int. Conference on Autonomous Agents*, 1998, pp. 47–53.
[32] R. Graham, "Bounds for certain multiprocessing anomalies," *Bell Systems Technical Journal*, vol. 45, pp. 1563–1581, 1966.

Peter Brass is professor of computer science at The City College of New York. He received his PhD in mathematics from the Technical University of Braunschweig, Germany, in 1992, and held a postdoc position in Greifswald, and a Heisenberg research fellowship of the german national science foundation before joining City College in 2002. He wrote books on 'Research Problems in Discrete Geometry' (with J. Pach, W. Moser, Springer 2005) and 'Advanced Data Structures' (Cambridge 2008). His current research interests are applications of algorithms, graph theory, and geometry in computational geometry, sensor networks, and robotics.

Flavio Cabrera-Mora is a PhD Candidate in Electrical Engineering at the Graduate Center of the City University of New York. He received his BS in Electrical Engineering from the Universidad Nacional de Colombia, Bogota, Colombia, in 1997, and his MS in Electrical Engineering from the City College of the City University of New York in 2006. His research interests include the areas of mobile robotics, multi-robot systems, and wireless sensor networks and its application for the exploration of unknown indoor environments.



Andrea Gasparri graduated cum laude in Computer Science in 2004 and obtained the Ph.D. degree in "Computer Science and Automation" in 2008, both from the University of Rome "Roma Tre", where he currently holds a postdoctoral position at the Department of Computer Science and Automation. His current research interests are in the areas of mobile robotics, sensor networks and more generally networked multi-agent systems.

Jizhong Xiao is associate professor of Electrical Engineering at the City College of New York (CCNY). He received his Ph.D. degree from Michigan State University in 2002; Master of Engineering degree from Nanyang Technological University, Singapore, and MS, BS degrees from East China Institute of Technology in 1999, 1993, and 1990, respectively. He started the robotics research program at CCNY in 2002 as the founding director of CCNY Robotics Lab. He is a recipient of USA National Science Foundation (NSF) CAREER award in 2007. His research interests include robotics and control, mobile sensor networks, robotic navigation, and multi-agent systems.