2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Multi-Robot Flooding Algorithm for the Exploration of Unknown Indoor Environments

Flavio Cabrera-Mora, Jizhong Xiao and Peter Brass

***Abstract*— In this paper we study the problem of multi-robot exploration of unknown indoor environments that are modeled as trees. Specifically, our approach consider that robots deploy and communicate with active landmarks in every intersection they encounter. We present a novel algorithm that is guaranteed to completely explore any tree with m edges and diameter D, by allowing k robots to be fed into the tree one at a time. We prove that the exploration time of the algorithm grows in linear proportion with the size of the tree and is not bigger than $D+m$. Simulation results are presented that corroborate the theoretical analysis.**

I. INTRODUCTION

The exploration of unknown indoor environments is a problem that remains open in the robotics community. Several approaches have been proposed to deal with this problem and can be mainly characterized by the way the environment is modeled: as a geometric structure [1], [2], [3]; as a grid [4], [5]; or as a graph [6], [7], [8], [9], [10].

In this paper we consider the exploration of an indoor environment with multiple robots. These robots are assisted by active landmarks (tokens) that are deployed by the robots while performing the exploration. These tokens provide directions to the robots that find them along the way. Considering these tokens as nodes of a graph and the passages between them as edges, the exploration process can be conveniently represented as the problem of exploring a graph: visiting all vertices and traversing all edges.

Exploration of graphs has been a well-studied problem, especially for the single-robot case, for both directed and undirected graphs. Exploration of directed graphs, where robots can move only in one direction, has been studied for example in [11] where the minimum time of exploration is investigated. In [12], it is proven that by letting a single robot to mark the environment with passive landmarks (pebbles) the exploration time is dramatically improved. In [13], it was shown that a single robot can learn its way through a directed graph in time polynomial in n, using only one pebble, if the upper bound on the total number of vertices n is known. If, on the other hand, this upper bound is not known, it is stated that $\theta(loglog(n))$ pebbles are both necessary and sufficient to perform the exploration. However, the authors in [13] do not mention how much the exploration time will be improved when using more than one pebble.

Flavio Cabrera-Mora is with the Department of Electrical Engineering, The Graduate Center of the City University of New York, 365 5th Ave. New York, NY, 10016 USA, fcabrera-mora@gc.cuny.edu

Jizhong Xiao and Peter Brass are with the Departments of Electrical Engineering and Computer Science, The City College of the City University of New York, 140th Street & Convent Avenue, New York, NY 10031, USA jxiao@ccny.cuny.edu, peter@cs.ccny.cuny.edu

Exploration using undirected graphs is considered in some other works. In [6] the goal is to achieve an efficient exploration under the constraint that the robot must return to the root for refueling periodically. In [14] the exploration of all vertices and all edges is achieved while minimizing the total number of edge traversals. In [15] and [16], undirected graphs were used to study the dynamic coverage problem using a single robot and the least recently visited (LRV) strategy. The robot deploys tokens that are capable of communication and sensing. These tokens assist the robot by providing instructions, although this instructions are limited to only the four cardinal directions. They proved that in a tree with m edges, the exploration time of LVR is $\leq 2m$, where their definition of complete exploration is that every edge has been traversed at least once.

Ideally, the exploration process must be accomplished as fast as possible and to this end, the use of multiple robots is expected to perform faster than the single robot case. In [17], an exploration algorithm that explores sparse trees with k robots is presented and it is shown that its competitive ratio is influenced only by the density and the diameter of the tree. In this approach, the communication exists exclusively among robots and it occurs only when robots meet in the same vertex. In [18], k robots are used to explore a tree with m edges and diameter D, using a communication model that allows all robots to exchange information at every step of the process through the use of tokens. The authors proposed an exploration algorithm with a running time $O(\frac{m}{\log k}+D)$, which is $O(\frac{k}{\log k})$ larger than the optimal exploration time when the tree is known a-priori. Finally, in [19], the authors presented an algorithm for the exploration of a tree with m edges and radius r, using k robots, and show that the exploration time is $\frac{2m}{k}+O(r^{k-1})$, improving the result in [18] and almost reaching the lower bound $max(\frac{2m}{k},2r)$.

Without exception all the aforementioned approaches for collective exploration assume that all k robots, located at the origin, enter the tree all at once in the first step. In practical implementations however, it will never be physically possible to make k robots enter an environment or traverse a passage all at once if the number of robots is large enough. Moreover, having k robots entering one particular edge all at once make them vulnerable to the environment. Furthermore, in some particular scenarios, such as a long path with no branches, the contribution of the k robots is null since all the robots will traverse the entire path, as if they were performing the exploration individually.

In this paper, we propose a novel algorithm in which robots enter an undirected tree one at a time and only one robot is allowed to traverse an edge at each step. On every vertex, robots deploy tokens that later on will instruct the robots toward unexplored/open edges, preventing robots to enter already explored subtrees. Consequently, the decision about the next move of a robot is mainly taken by the tokens. This creates a decentralized system, freeing robots resources for other tasks. We derive an upper bound for the exploration time, proving that it depends linearly on the size of the tree and that is never bigger than $D + m$. We also perform simulation to study the behavior of the algorithm in trees of different sizes.

The paper is organized as follows: Section II describes our proposed flooding algorithm. Its analysis on trees is performed in Section III. Simulations showing the behavior of the algorithm in different trees are presented in Section IV. Finally,concluding remarks are offered in Section V.

II. Flooding Algorithm

A. Preliminaries

k Robots are initially located at the root r of an unknown tree $T = (V, E)$, containing n vertices and m edges. Each robot is uniquely identified with an integer number from 1 to k. Robots explore the tree by moving along the edges that link a pair of vertices. All robots are identical and explore the tree following the same exploration algorithm. Define a parent vertex as one with at least one edge leading to another vertex in a downward direction. The latter vertex is called a child vertex. A leaf is a vertex with no children. Define a downward edge $\check{e}_v$ as an edge that leads from a vertex v to one of its children. That is, for any vertex v, the number of downward edges can be expressed as: $|\check{e}_v| = deg(v) - 1$.

Define a token (active landmark) as a bookkeeping device being deployed in each vertex. Every token has communication capabilities and is able to store at least the following information:

- The number of edges converging in a vertex
- The ID of the robots that have visited it
- A list of available edges for robots to explore
- The direction that a robot will take in the next step

Each robot is capable of:

- Carrying an unlimited number of tokens.
- Assessing whether it has reached a vertex and deploy a token if none is present in it.
- Distinctively identifying every edge on that vertex.

At every step, a robot located at a vertex can either traverse an edge or remain in its current position. Define an unused/unexplored edge as one that still has not been traversed by any robot. Define an open edge as one through which a robot has left a vertex v, without returning back to it nor sending a message that it has reached a leaf (i.e., the robot is still exploring downwards the subtree rooted at v). If a vertex has some open edges, the token will send a robot to each open edge in increasing order. For instance, let assume vertex v has open edges labeled 2, 5, 8 and 10. The next robot arriving at v will be sent to edge 2 the next step. Any robot that arrives at v later will be sent to edge 5, and so forth, until one robot has been sent to each open edge distributing the robots as evenly as possible.

Assume that at each step, each token is able to transmit its own information about available edges to its parent token and that this transmission is instantaneous. At the beginning, the first robot deploys a token at r. Each time a robot reaches a vertex where a token has been deployed before, it will read and write its information. Define back-pointer as a list of all edges a robot has visited in order to be in its current position. Each edge in the list must have associated the ID of the two tokens the edge is connected to. This list can be used later on by the robot to back trace its steps to r. A robot arriving at a leaf will drop a token, marking the place, and the token will transmit to its parent that there are no branches available at this end. This will prevent robots to enter the edge again and will inform that the robot currently located at the leaf is returning back to its parent in the next step.

B. "One-by-one" Flooding Algorithm

The main difference between our algorithm and those in the literature is that ours allows robots to enter the tree one at a time and only one robot is allowed to traverse an edge in each step. In addition, each token exchanges information with its parent about the next destination of the robot in the vertex. As a result, no two robots will enter the same vertex on the same step.

Algorithm 1: "One-by-one" Flooding Algorithm

```
1  Let $R$ be a robot arriving at a vertex $v$ through edge $e$;
2  Let $e_j$ be the ID of each edge at $v$;
3  Let $j$ be the total number of edges at $v$;
4  if there is no token present at $v$ then
5      $R$ drops one token at $v$;
6      $R$ follows one of the $j$ available edges in next step;
7  else if there are unused edges at $v$ then
8      $R$ follows one of the unused edges in next step;
9  else if there are open edges at $v$ then
10     if open edges to be used by other robots in next
       step then
11         $R$ will stay put at $v$ in next step;
12     else
13         $R$ follows one open edge in next step;
14     end
15 else
16     $R$ returns to the parent vertex of $v$ using its
       back-pointer in next step;
17 end
```

The algorithm exhibits four properties:

1) No edge is used more than twice by the same robot.
2) Some robots might remain static in some vertices (they will not enter an edge unless an instruction to do so is given by the token at the vertex).
3) The exploration is performed by the number of robots that is more appropriate for the size of the tree.

4) No robot returns to r before every edge has been traverse and every vertex has been visited at least once.

III. Result and Analysis on Trees

Define *finished tree* as a tree in which all edges have been traversed and all vertices have been visited at least once by any robot, and the *finished exploration time (t_e)* as the time needed for the tree to be finished. Define *completed tree* as a tree that is finished and in which all robots are back at r. The *complete exploration time (t_c)* is the time required for the tree to be completed.

Consider a tree T with m edges, rooted at vertex r and define d_r as the degree of r. Assume that there are k robots stationed at r and that it is guaranteed that there will be robots available at all times in r to be fed into the tree until the exploration is finished.

Claim 1: Any tree can be considered as being formed by d_r individual subtrees, all of them rooted at r, where the degree of each root is equal to 1.

Proof: By definition, the algorithm will feed one robot into each available unused/open edge at every step. At the start of the exploration, there are d_r unused edges in r, along with k robots. The first d_r robots will enter one of the d_r unused edges in the first step since there is nothing that prevent them to do so (recall that only a token can instruct the robot to remain static in a vertex). The feeding process starts at the same time on all the subtrees of r. Whenever a subtree is finished, the robots in that subtree will start returning to r, but this will not affect the movement of robots in the other subtrees of r. Thus, the exploration of a subtree of r is independent of the exploration of any other subtree, and as such every subtree can be thought of as an individual tree where its root has degree 1. ■

Define a path $S = [a_0, a_1, a_2, \dots, a_{l-1}, a_l, a_{l+1}]$ as the path from the root to the vertex that will be explored last, where $a_0 = r$ and the last element is a leaf. Define T_s as the subtree of r where S is specified. Since we are interested in the worst performance, at every vertex the decision of what edge a robot should take next step belongs to the adversary. That is, if vertex a_i has j unexplored edges, the edge that leads to a_{i+1} will be taken after all other $j-1$ edges has been taken before.

Claim 2: The *complete exploration time* of tree T equals the time needed to completely explore the subtree T_s.

Proof: Proof of claim 2 follows directly from claim 1: since each subtree of r is independent of each other, and because the subtree T_s is the subtree that will be completely explored last, any other subtree will be completed before T_s. Consequently, T_s will determine the time in which tree T is completely explored. ■

Due to claim 2, from now on the analysis of tree T will be referred as the analysis of T_s.

Claim 3: The total time to complete the exploration t_c equals the total time to finish the exploration t_e plus the number of robots k_m inside the tree at time t_e. That is,

$$t_c = t_e + k_m \qquad (1)$$

Proof: One of the properties of the algorithm states that all robots that enter the tree will remain in it until the exploration is finished either by moving along open/unexplored edges, or by remaining static in a vertex. If at time t_e there are k_m robots inside the tree, and since only one robot is allowed to traverse an edge in each step, then it will take k_m steps to evacuate all robots to r. ■

Claim 4: With algorithm "one-by-one", the maximum number of robots entering a tree equals t_e. That is,

$$k_{m_{max}} = t_e \qquad (2)$$

Proof: According to the algorithm, at each step one robot enters a tree. If conditions are adequate, the algorithm will keep feeding robots until time t_e is reached. Since t_e is the time when the last unexplored edge is traversed, after t_e all robots in T will start returning to r. ■

Theorem 1: Algorithm "one-by-one" will completely explore any tree in time at most $D + m$. That is,

$$t_c \leq D + m \qquad (3)$$

Proof: In the first part of the proof we will consider how the algorithm perform in smaller trees. Define *"minimum tree"* as a 3-edge tree with $D = 2$ and $m = 3$ that will always be explored by two robots in time $t_e = 3$ (Fig. 1). As a result, the time for complete exploration will be: $t_c = t_e + k_m = 5$. Note that t_c can also be defined, for this configuration, as: $t_c = D + m = 5$. Define *"extended minimum tree"* as a tree with $D = 2$ and $m_e = j + 1$ edges, where j is the total number of leaves. The extended minimum tree will also be explored by at most two robots (Fig. 2). In general, the exploration time of any extended minimum tree with two robots can be expressed as:

$$t_e = m_e \qquad (4)$$

$$t_c = t_e + k_m = m_e + 2 = m_e + D \qquad (5)$$

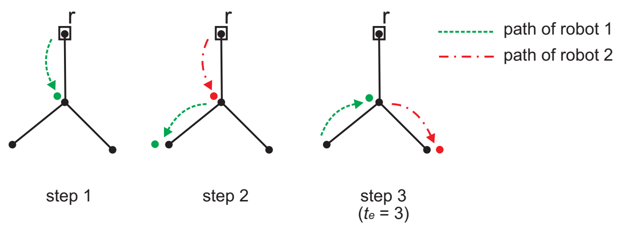

Fig. 1. Two robots exploring the *"minimum tree"*

Now, consider a tree with diameter $D \geq 3$. According to the definition of internal vertex, the tree with the minimum number of edges and diameter $D \geq 3$ has the configuration shown in Fig. 3a. Fig. 3b shows that this tree can be decomposed into $D - 1$ minimum trees. Since each minimum tree will be explored in $t'_e = m'_e$ with two robots, then the total

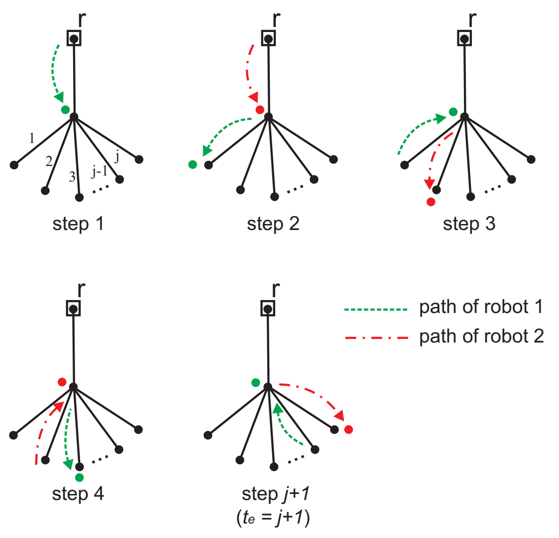

Fig. 2. Two robots exploring an *"extended minimum tree"*

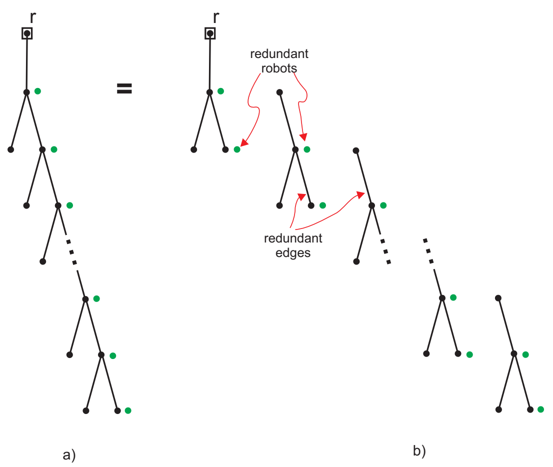

Fig. 3. a) Tree with minimum number of internal vertices and $D \geq 3$ (dots represent robots located at vertices at time t_e) b) Decomposition into $D-1$ minimum trees

exploration time will be equal to $t_e = (D-1)m'_e$. However, this result is inaccurate: there are some redundant edges, as shown in Fig. 3b. For each $D-1$ minimum trees forming T there will be $D-2$ redundant terms. Consequently, the total exploration time can be expressed as,

$$t_e = (D-1)m'_e - (D-2) \tag{6}$$

The total number of edges in T can be obtained from the following analysis: each minimum tree has m'_e edges, and there are $D-1$ minimum trees and $D-2$ redundant edges that belong to more than one minimum tree. Thus, the total number of edges in T can be expressed as,

$$m = (D-1)m'_e - (D-2) \tag{7}$$

Comparison of (6) and (7) leads to: $t_e = m$.

The number of robots inside the tree in Fig. 3a at time t_e can be obtained following a similar analysis: each minimum trees is explored by two robots. There are some robots that are considered in more than one minimum tree (cf. Fig. 3b). That is, for each pair of minimum trees, there is one redundant robot. Thus, the total number of robots at time t_e can be expressed as:

$$k_m = 2(D-1) - (D-2) = D \tag{8}$$

Using claim 3, we can express t_c as,

$$t_c = t_e + k_m = m + D \tag{9}$$

Thus, theorem 1 holds for any small tree of diameter D and with the minimum number of edges in it.

In the second part of the proof, we study the behavior of the algorithm in any larger tree. To this end, we will use a full N-ary tree. A full N-ary tree is one in which all leaves have the same depth (diameter) and all internal vertices have degree N, where by definition $N \geq 2$ and $D \gg N$. Consider there exists a N-ary tree rooted at a_1 with diameter h. The path S is defined in the N^{th} subtree as shown in Fig. 4.

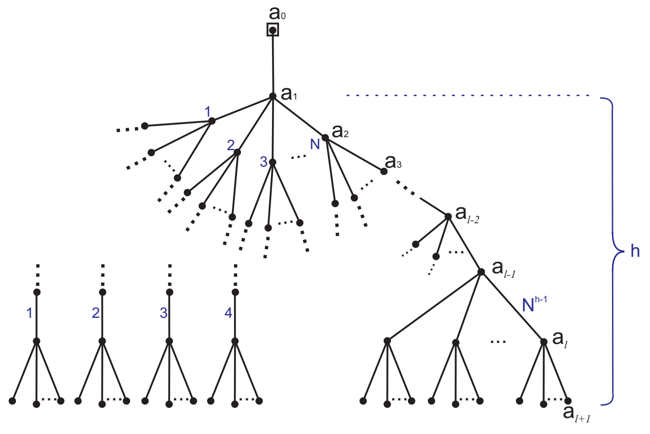

Fig. 4. N-ary tree rooted at a_1

The total number of leaves in a N-ary tree equals N^h, and the number of internal vertices can be calculated as follows,

$$\begin{aligned} 1 + N + N^2 + \cdots + N^{h-1} &= \sum_{i=0}^{h-1} N^i \\ &= \frac{N^h - 1}{N-1} \end{aligned} \tag{10}$$

Since it is assumed that the N-ary tree is rooted at a_1, the total number of vertices n in T equals,

$$n = \frac{N^h - 1}{N-1} + N^h + 1 \tag{11}$$

Where the one accounts for vertex a_0. The total number of edges in T is,

$$m = n - 1 = \frac{N^h - 1}{N-1} + N^h = \frac{N^{h+1} - 1}{N-1} \tag{12}$$

Since $h = D-1$, then (12) can be rewritten as,

$$m = \frac{N^D - 1}{N-1} = N^{D-1} + N^{D-2} + \cdots + N + 1 \tag{13}$$

If the *"one-by-one"* algorithm is used in the N-ary tree, the first subtree of a_1 will be finished first, then the second subtree and so forth. Once finished, a subtree will be blocked for other robots to enter. By time t_e, it is likely that some robots have returned to a_1 and entered any other open subtree. This will block robots at r to enter the tree at every step, and as a result the total number of robots in the tree at time t_e will be less than the maximum possible number of robots (recall claim 2). That is,

$$k_m < t_e \tag{14}$$

As a result,

$$\begin{aligned} t_c &= t_e + k_m \\ &< 2t_e \end{aligned} \tag{15}$$

Now we can look for t_e: if the N-ary tree has in total N^h leaves, there are N^{h-1} "extended minimum trees" in T. Assume that all extended minimum trees can be labeled with numbers from 1 to N^{h-1}, where the latter corresponds to the one rooted at a_{l-1} that leads to the last vertex in the path S. Assume that all the N^{h-1} extended minimum trees will be explored by only one robot. This will guarantee that t_c is maximized. Additionally, any extended minimum tree will be entered by one robot after the robot has traversed at least $h-2$ steps (i.e., the time it takes to go from a_1 to the root of the last extended minimum tree of the subtree). Finally, the first robot to enter the $N^{(h-1)th}$ extended minimum tree will do so after a robot has been sent to any other extended minimum tree. That is, the first robot will enter $N^{(h-1)th}$ extended minimum tree after $N^{(h-1)}-1$ steps have passed. All these times are with reference to the N-ary tree which is rooted at a_1. In order to refer them to a_0 we need to add one additional step. As a result, the edge connecting a_l to a_{l+1} will be traversed at time,

$$\begin{aligned} t_e &= (h-2) + 2N + (N^{h-1}-1) + 1 \\ &= N^{h-1} + 2N + h - 2 \end{aligned} \tag{16}$$

Using (16) into (15),

$$\begin{aligned} t_c &< 2(N^{h-1} + 2N + h - 2) \\ &< 2N^{h-1} + 4N + 2h - 4 \end{aligned} \tag{17}$$

Expressing t_c as a function of D yields,

$$t_c < 2N^{D-2} + 4N + 2D - 6 \tag{18}$$

On the other hand, the sum $D+m$ results on,

$$\begin{aligned} D+m &= D + \frac{N^D - 1}{N-1} \\ &= N^{D-1} + N^{D-2} + \cdots + N + 1 + D \end{aligned} \tag{19}$$

Comparison of (18) and (19) reveals that $t_c < D+m$. Thus, theorem 1 holds for any large tree.

Finally, combining both parts of the proof yields that for any tree $t_c \le D+m$. ■

IV. Simulation Results

We simulate the behavior of our *"one-by-one"* flooding algorithm in different trees of increasing size (increasing number of vertices n), starting with the *"minimum tree"* (n=4). Simulation results show the upper bound of exploration for every tree and the *complete exploration time* given by the algorithm. A typical run for trees of up to 200 vertices is show in Fig. 5. The algorithm always explores the tree in its entirety. Fig. 6 shows that for a given number of

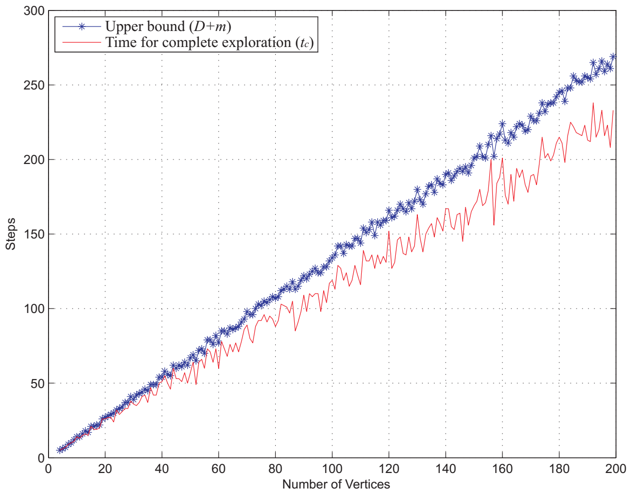

Fig. 5. Simulation results for a typical run of the algorithm in trees of different sizes from n=4 to n=200

vertices different configuration of trees are possible, where every configuration will have different exploration times.Fig. 7 shows the simulation results for 50 randomly generated tree configurations per each n ranging from 4 to 100. The complete exploration time corresponds to the mean value of the 50 runs along with its standard deviation per each value of n. The result of the simulations (Figs. 5 and 7) show that

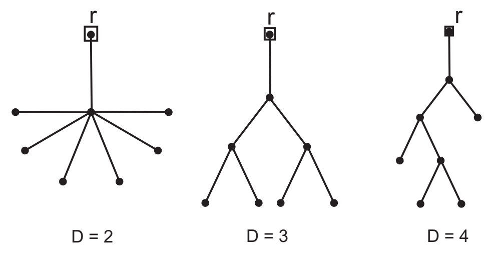

Fig. 6. Example of different tree configurations for the same number of vertices (n=8)

t_c is always less or equal than the proposed upper bound. In fact, as the size of the tree grows the gap between the upper bound and t_c increases, guaranteeing that the upper bound will never be violated even if the size of the tree goes to infinite. Another interesting result of the simulations is that the grow of t_c is almost linear with the size of the tree. This is a very desirable characteristic since it shows its predictable nature. Furthermore, Fig. 7 shows that the

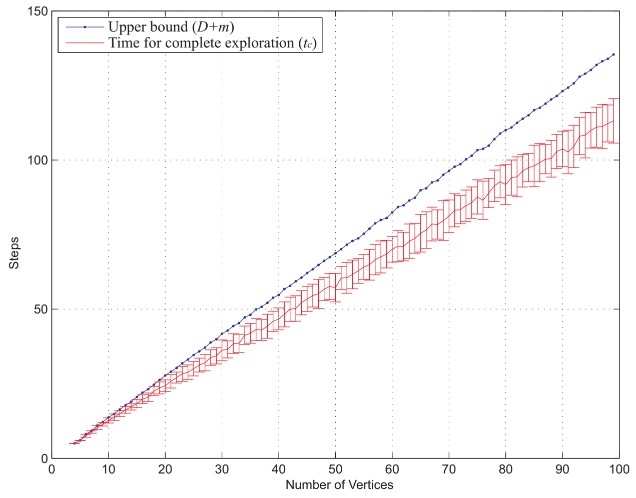

Fig. 7. Simulation results for 50 runs of the algorithm in trees of different sizes from n=4 to n=100

proposed upper bound is tight with respect to the complete exploration time of the algorithm. Finally, Fig. 8 shows that the relationship between the number of robots k needed to explore an unknown tree and the size of the tree is almost linear. This is an important fact about the algorithm since it glimpses the existence of a linear upper bound in the number of robots needed to perform the exploration of any unknown tree with our algorithm. That is, there exists a limit in the number of robots beyond which the exploration time will not be improved.

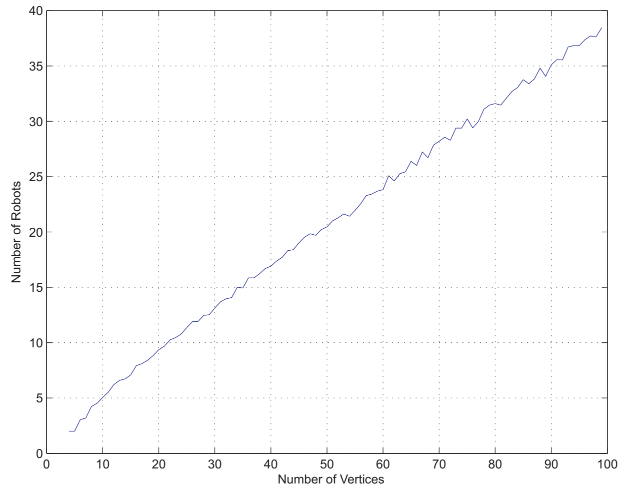

Fig. 8. Number of robots used for completely explore trees of different sizes from n=4 to n=100 using "one-by-one" algorithm

V. Conclusions

In this paper, we present a novel algorithm that is guaranteed to completely explore any undirected tree with m edges and diameter D. The algorithm differs from other works in the literature by the fact that robots are allowed to enter only one at a time and that edges can be traversed by only one robot in each step. We prove that the complete exploration time of the algorithm t_c is never larger than $D + m$. This proposed upper bound is independent of the number of robots k needed to perform the exploration, another difference with respect to other approaches in the literature. We simulate the behavior of the algorithm on trees of different sizes and show that the analysis of section III produces a tight upper bound with respect to the complete exploration time. The simulations also show that both, the growth of t_c and the number of robots k needed to perform the exploration are linearly related to the size of the tree.

References

[1] 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.

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

[3] 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.

[4] B. Yamauchi, "Frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence, Robotics and Automation*, Monterrey, CA (USA), July 1997, pp. 146–151.

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

[6] 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.

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

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

[9] A. Dessmark and A. Pelc, "Optimal graph exploration without good maps," *ESA 2002, ser. LNCS*, vol. 2461, pp. 374–386, 2002.

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

[11] 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.

[12] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859–865, December 1991.

[13] 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)*, Dallas, TX (USA), May 1998, pp. 269–278.

[14] P. Panaite and A. Pelc, "Exploring unknown undirected graphs," in *Proceedings of the ninth annual ACM symposium on Discrete Algorithms (SODA)*, San Francisco, CA (USA), January 1998, pp. 316–322.

[15] M. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, vol. 26, no. 2-4, pp. 181–196, June 2004.

[16] ——, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661–675, August 2007.

[17] M. Dynia, J. Kutylowski, F. M. auf der Heide, and C. Schindelhauer, "Smart robot teams exploring sparse trees," *Lecture Notes in Computer Science*, vol. 4162, pp. 327–338, August 2006.

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

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