Maintaining Communication in Multi-Robot Tree Coverage *

Mor Sinay [1], Noa Agmon[1], Oleg Maksimov[1], Sarit Kraus[1], David Peleg[2]
[1]Bar-Ilan University, Israel
[2] The Weizmann Institute, Israel
mor.sinay@gmail.com, {agmon,sarit}@cs.biu.ac.il,
oleg@maksimov.co.il, david.peleg@weizmann.ac.il

Abstract

Area coverage is an important task for mobile robots, mainly due to its applicability in many domains, such as search and rescue. In this paper we study the problem of multi-robot coverage, in which the robots must obey a strong communication restriction: they should maintain connectivity between teammates throughout the coverage. We formally describe the *Multi-Robot Connected Tree Coverage* problem, and an algorithm for covering perfect N-ary trees while adhering to the communication requirement. The algorithm is analyzed theoretically, providing guarantees for coverage time by the notion of *speedup factor*. We enhance the theoretically-proven solution with a *dripping* heuristic algorithm, and show in extensive simulations that it significantly decreases the coverage time. The algorithm is then adjusted to general (not necessarily perfect) N-ary trees and additional experiments prove its efficiency. Furthermore, we show the use of our solution in a simulated office-building scenario. Finally, we deploy our algorithm on real robots in a real office building setting, showing efficient coverage time in practice.

1 Introduction

A popular application of mobile robots is coverage: visiting each location in a known or unknown environment in order to perform a task [Rogge and Aeyels, 2007a; 2007b; Hazon and Kaminka, 2008; Jensen and Gini, 2013; Jensen *et al.*, 2014]. The problem has been studied extensively using a single robot, seeking a coverage path that visits each point in the environment at least once in minimal time, e.g., [Gabriely and Rimon, 2001]. Naturally, one can speed up the coverage using multiple robots. In the multi-robot coverage problem, the goal is to compute a trajectory for each robot in the team so that the maximal coverage time (that is, the longest travel time of any robot) is minimized among all robots.

One popular approach is to look at the coverage problem as a problem of covering a graph $G = (V, E)$ [Rogge and Aeyels, 2007a; 2007b; Jensen and Gini, 2013; Jensen *et al.*, 2014]. Another approach is to consider the coverage problem of a tree $T = (V, E)$ [Fraigniaud *et al.*, 2004; Brass *et al.*, 2011; Cabrera-Mora and Xiao, 2012]. Under this representation, at each time step, it should be decided for each robot from the team which neighboring node it should visit. Thus, the goal is to visit all nodes of the graph, at least once, as quickly as possible.

In this paper we examine the problem of covering a perfect N-ary tree (that is, a rooted tree in which each node—except for the leaves—has exactly N children) by a team of robots while maintaining communication between the robots, when the tree is known in advance. The robots are located on the nodes of the tree and can move simultaneously along the edges. Two robots are considered to be in communication range if there is an edge between the nodes on which they are located. A tree environment is a convenient form of representing disaster areas, where there is only one path to reach any point on any specific location, thus there is only one path between any pair of nodes [Fraigniaud *et al.*, 2004; Brass *et al.*, 2011; Cabrera-Mora and Xiao, 2012].

Communication-constrained coverage problems are not new, and exist in the literature. However, these solutions either do not present theoretical analysis of coverage time, or use active landmarks (or similar) to coordinate the robots' movements. In this paper we present the N-ary Connected Coverage Tree Algorithm (NCOCTA) for covering a given perfect N-ary tree by a team of k robots without using any external devices. We provide a theoretical analysis of the coverage time using the notion of *speedup factor* ($\mathsf{SF}(\mathcal{A})$) [Wilkinson and Allen, 1999], which represents the speedup attained by some algorithm $\mathcal{A}$ using k robots compared to the optimal coverage time achieved by a single robot. We enhance the theoretically-proven NCOCTA algorithm by using a *dripping* heuristic algorithm, the Connected Coverage Tree Algorithm (COCTA), that was shown in extensive simulations to significantly decrease the coverage time of NCOCTA. In addition, the COCTA algorithm works on general trees. We have implemented our solutions on ROS/Gazebo [1], a realistic robotic simulation, and deployed our solution on real robots, demonstrating the efficiency of our coverage algorithms in a real office building setting in practice.

*This research was supported in part by a grant from the Ministry of Science & Technology, Israel & the Japan Science and Technology Agency (jst), Japan & ISF grant #1337/15.

[1] http://www.ros.org, http://gazebosim.org

2 Related Work

Robotic coverage[2] is a canonical problem in robotics, which has received considerable attention in the literature. The single-robot coverage problem can be solved optimally in polynomial time under the assumption that the environment is represented as a grid, using the Spanning Tree Coverage (STC) algorithm [Gabriely and Rimon, 2001], where the robot follows a spanning tree over the grid.

Fraigniaud et al. [2004] and Brass et al. [2011] developed online algorithms for exploring unknown trees and graphs, respectively. Fraigniaud et al. [2004] assumed that the robots can communicate by writing the acquired information in the node currently being visited, and reading the information available at this node. They proved a competitive ratio for k robots of $O(k/\log k)$ for the time of exploration of an unknown tree compared to the time of an optimal algorithm which knows the tree in advance. Brass et al [2011] proposed an algorithm for exploring an unknown graph and returning to the starting point. Forster and Wattenhofer [2016] obtained upper and lower bounds for the competitive ratio of online exploration of directed and weighted graphs. Unlike their approach, we assume the tree is given in advance and that the robots can communicate explicitly, but in a limited range. Cabrera-Mora [2012] proposed a multi-robot exploration algorithm of a known tree using landmarks that aim at minimizing the time of exploration but take into account the overall distance traversed. The robots coordinate their movements in a decentralized manner, relaying the information stored in the active landmarks. They obtain upper and lower bounds on the coverage time, but their model allows a vertex to be occupied by only one robot and an edge to be traversed by only one robot at each time step. Pei and Mutka [2012] presented an algorithm for exploring an unknown environment. They considered the problem of minimum path finding for the relay-deployment robot to travel and the positions to deploy necessary relays to support the stream aggregation in each movement iteration. We assume that there's no bandwidth problem.

Similar to our communication assumptions, Jensen and Gini [2013], Jensen et al [2014] and Rogge and Aeyels [2007b; 2007a] developed algorithms for exploring a terrain modeled as a graph. Jensen and Gini [2013] proposed a Rolling Dispersion Algorithm (RDA) for exploring an unknown area. The robots disperse as much as possible while maintaining wireless communication and then advance as a group, leaving behind beacons to mark explored areas and provide a path back to the starting point. Jensen et al [2014] proposed the Sweep Exploration Algorithm (SEA), which uses a much more restrictive communication model, thus one robot at a time travels down a single path until it is completely explored, then it retracts and explores a new path. Rogge and Aeyels [2007b; 2007a] developed a coverage algorithm for unknown terrains. They assumed that communication between the robots is restricted to line-of-sight and to a maximum distance. Each robot explores a different part of the unknown region and sends its findings to a central device which combines the data received from the robots into one global map of the area. Banfi et al [2016] consider the communication constraints for the case in which robots must connect to a base station only when new information is collected, allowing robots to be disconnected for arbitrarily long periods. In these papers, only completeness guarantees are provided, and coverage time is evaluated empirically, while we evaluate the performance (exploration time) theoretically.

[2]The notion of multi-robot coverage and exploration are interchangeable, and both can be found in the literature.

3 The Connected Tree Coverage Problem

Consider an environment that is mapped as a perfect N-ary tree $T = (V, E)$ where $V = \{v_i\}$ is the set of graph nodes and $E = \{e_{ij}\}$ is the set of edges, and an edge $e_{ij} = (v_i, v_j)$ exists if a robot can move directly from v_i to v_j, $v_i, v_j \in V$. A direct connection between two nodes exists if robots can communicate when located on the nodes, and along the edge between them. A simple interpretation is line-of-sight. Let H be the height of tree T (the leaves are at height 0), and k be the number of robots in the team. Robots r_i and r_j, located in nodes $v_{i'}, v_{j'}$ respectively, are said to be in *communication range* in T if $e_{i'j'} \in E$, or if $v_{i'} = v_{j'}$ (i.e., they are located in the same vertex). Thus, in general, a team of $k \geq 2$ robots is said to be connected at time t if the subtree induced by the locations of the robots forms one connected component.

The *Multi-Robot Connected Tree Coverage* (MRCTC) Problem is defined as follows: Given a perfect N-ary tree $T = (V, E)$ of height H, and a team $R = \{r_0, \ldots, r_{k-1}\}$ of robots initially located in the root of T, find a coverage path for each robot $r_i \in R$ such that each $v_i \in V$ is visited by some robot, all robots in R are connected throughout the coverage, and the total coverage time is minimized.

Unfortunately, the MRCTC problem is **NP**-hard, based on the analysis in [Fraigniaud *et al.*, 2004], which shows that the collective tree exploration is **NP**-hard even if the tree and the starting node are known in advance. We therefore turn to a solution that has a theoretically proven speedup factor.

We propose a novel strategy to solve the MRCTC problem on perfect N-ary trees: NCOCTA. An important aspect of NCOCTA is where the robots split into subtrees (leaving at least one robot at the location of the split, to maintain connectivity between the subtrees). In particular, the parameter of how many points along the exploration (height in the tree) they split into plays an important role. NCOCTA guarantees that SF(NCOCTA) $= 2N^m - 1$, where $m = \arg\max_m\{1+N^m(\log_N(m)+m+4)+(N^m-N)\frac{1}{N-1} \leq k\}$ (m represents the number of heights where the robots split, as presented in section 4.1). We then describe the heuristic algorithm COCTA, a modification to NCOCTA, which allows the robots to "drip" to a neighboring subtree without waiting for all of the robots to finish the current subtree coverage. COCTA is proven to have SF(COCTA) $\geq$ SF(NCOCTA), and rigorous empirical evaluation shows that in practice SF(COCTA) $>>$ SF(NCOCTA).

4 Solving MRCTC on Perfect N-ary Trees

In this section we describe in detail the solution to the MRCTC problem for perfect N-ary trees. The algorithm

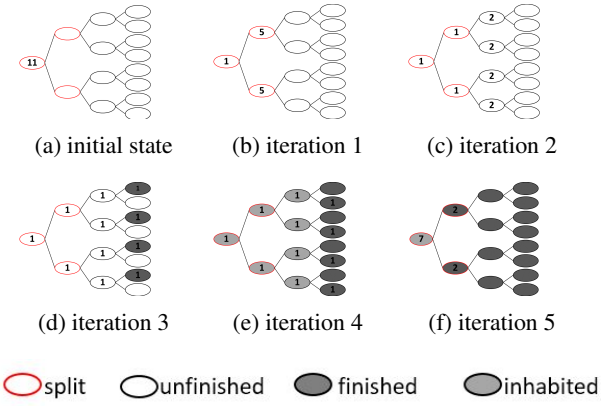


Figure 1: Simulation of NCOCTA algorithm on a perfect 2-ary tree when $k = 11$, $H = 3$ and $h = \{3, 2\}$. The number in a node represents the number of agents located in that node.

seeks to find the best way to *split* the given k robots between different subtrees under the communication restriction. The height of the first split (starting from the root and going down) defines the first location in which one robot should be left in order to maintain communication between the N subtrees. Following this split, the exploring robots leave a trail of robots along the coverage path, to guarantee that the team will not disconnect. See Figure 1 for a demonstration of a split (note that it does not present an *optimal* split). Analayzing the theoretical SF, we will first assume that we are given k robots and the heights of the splits $\{h_i\}_{i=1}^m$. After finding the SF formulation, we will find the best heights to split the robots in order to gain the best SF for any given k.

4.1 NCOCTA Algorithm

In this subsection we will describe the NCOCTA algorithm, whose speedup factor will be analyzed in Section 4.2. In this algorithm, we are given a set of predefined heights $\{h_i\}_{i=1}^m$ at which the robots will stop moving together along the tree, and are split between N subtrees rooted at this height. In order to maintain communication along the trail of the robots to all splits we require that $\forall 1 \leq i \leq m$, $H \geq h_i > h_{i+1}$ and that k be at least $1 + Nh_1 + \sum_{i=2}^{m}(N^i - N^{i-1})h_i$.

We denote by T_u a subtree of an explored tree T, rooted at node u. The node u is *finished* if T_u is explored and either there are no robots in it, or all robots in it are in u. Otherwise, it is called *unfinished*. Node u is *inhabited* if there is at least one robot in T_u. Denote by $h(u)$ the height of node $u \in V$.

A demonstration of several iterations of the NCOCTA algorithm on a perfect 2-ary tree with height 3 ($H = 3$) and a set of splits $h_1 = 3, h_2 = 2$ using 11 robots ($k = 1+2\cdot3+2\cdot2$) is presented in Figure 1.

4.2 Properties of the NCOCTA algorithm

Lemma 4.1. *During the execution of* NCOCTA*, the robots maintain communication among themselves, hence the induced subtree creates one connected component.*

Proof. Initially, all robots are in the root of the tree, so the claim is trivially true. Assume, in contradiction, that there is an iteration of the algorithm that changes the induced subtree (of height h) from one connected component to several connected components. If the robots are located on a node u of height $h(u) > h_1$, then all the robots move together (by the initial split rule described above), thus all k robots are located in the same node, which is equivalent to the initial state (so we can exclude this case and assume $h(u) \leq h_1$).

Since we have two (or more) connected components, there was a node u from which the robot separated into two components. This separation can happen in one of two cases: (1) a subteam of R moved down a subtree of u (without leaving a representative in a node u' which is a child of u) while a subteam of R remained in u; or (2) a subteam of R went up the tree (without leaving a representative at u'', the parent of u).

In the first case, moving to a child will occur when T_u is unfinished. We leave a robot behind for any case except $h(u) > h_1$ (in that case all k robots are located in u and

Algorithm 1 NCOCTA

```
while root is not finished do
  for all v ∈ V in which robots are located on (go over from the
  lowest to the highest height) do
    if v is finished then
      \* There are no robots in T_v subtrees, so we can return
      to the parent node *\
      if v ≠ root then
        All robots from v go to the parent of v.
      else
        All robots from v stop.
      end if
    else if ∃ u a child of v such that u is unfinished then
      \* v is unfinished, hence, there are still nodes to explore.
      We can move to a child*\
      if ∃h_i such that the h(v) = h_i then
        Leave one robot at v and split the rest equally among
        the children.
      else if h(v) ≤ h_1 then
        Select a child u of v such that u is unfinished.
        Move all the robots in v to u leaving one robot in v.
      else
        Select a child u of v such that u is unfinished.
        Move all the robots in v to u.
      end if
    else
      \* v is inhabited, the subtree is explored but there are still
      robots in the subtree. The robots wait on a node (vertex)
      with height ∈ {h_i}_{i=1}^m (split node) until all the robots
      arrive. Now, the node is finished and the robots move
      together to another subtree. *\
      if ∃h_i such that the h(v) = h_i then
        All robots from v remain in v.
      else if v ≠ root then
        All robots from v go to the parent of v leaving one
        robot in v.
      else
        All robots from v stop.
      end if
    end if
  end for
end while
```

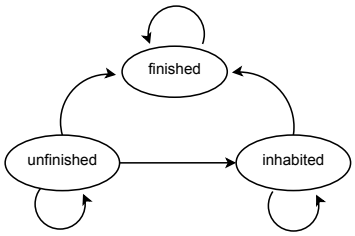


Figure 2: A DAG representing the possible states of each $u \in V$

move together), leading to a contradiction.

In the second case, moving to a parent while not leaving a robot behind can occur only if T_u is finished, but if T_u is finished, there are no robots in any node of the subtree except for u, again leading to a contradiction. □

Lemma 4.2. *Algorithm* NCOCTA *completes exploring all nodes $v \in V$ in finite time.*

Proof. For every $v \in V$, v can be finished, unfinished or inhabited. The directed acyclic graph, or DAG (except for self loops) representing the possible states of v is presented in Figure 2. The initial state of all $v \in V$ is *unfinished*, and the desired state of all nodes is *finished*. If we show that for a given tree, after a finite time (δ_t) the state of at least one $v_i \in V$ will change (according to the DAG described in Figure 2), then after at most $2|V|\delta_t$ time units, all nodes in V will reach the final state (*finished*).

There are two cases to consider: $[A]$ there is at least one node v_i which is unfinished, and v_i contains robots. $[B]$ every node v_j with robots is either finished or inhabited (meaning, we do not have any $v_l \in V$ which is unfinished, with robots located on v_l).

$[A]$ Let us consider the node v_i with the minimum height that has robots located on it. Following the condition on k (the number of robots k is at least $1+Nh_1+\sum_{i=2}^{m}(N^i-N^{i-1})h_i$ as shown in section 4.1), if v_i is unfinished and $h(v_i) \leq h_1$, then we have at least one robot which can move to a child of v_i while maintaining communication, hence it continues moving to a child recursively until reaching a leaf, thus the leaf changes to *finished* within $h(v_i)$ time units, and we are done (meaning, $\delta_t = h(v_i)$). If $h(v_i) > h_1$, then all robots move together until they reach a node of height h_1, continuing as shown previously.

$[B]$ There exists a node v_j of the lowest height on which robots are located, and v_j is finished. We know that v_j's parent is inhabited (otherwise it would have been case [A]). If v_j's parent is not a split node (a node u where $h(u) \in \{h_i\}_{i=1}^{m}$), then all robots go up to v_j's parent and v_j's parent becomes *finished*, thus within $\delta_t = 1$, v_j's parent's state has changed. If v_j's parent is a split node, then since T is a perfect N-ary tree and by NCOCTA all movements of robots along the paths occur simultaneously, then when going up to v_j's parent, all subtrees which originated in the parent move back to it simultaneously, thus v_j's parent's state changes to *finished* within $\delta_t = 1$. □

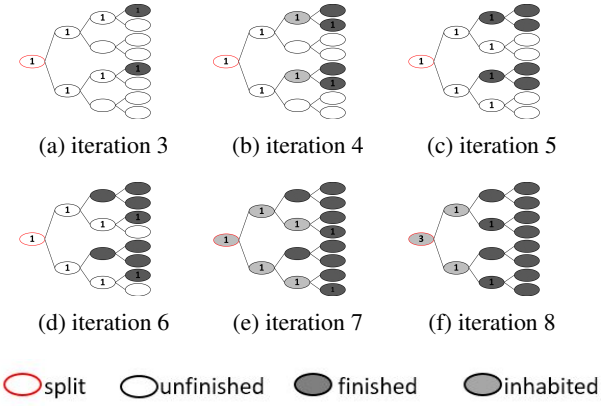


Figure 3: Simulation of NCOCTA algorithm when $k = 7$, $H = 3$ and $h = \{3\}$. The number in a node represents the number of agents located in that node.

We turn to analyze the NCOCTA algorithm using the notion of *speedup factor*, i.e., the difference between the coverage time guaranteed by NCOCTA, compared to the optimal coverage time of one robot. For clarity, the proofs refer to the simpler case of perfect 2-ary trees (full binary trees), and the generalization for perfect N-ary trees, $N \geq 2$, is described briefly. We start by providing some mathematical foundations for the proof. Recall that we are given a set of heights $\{h_1, \ldots, h_m\}$ in which the robots split equally between subtrees of nodes $\{v_i | h(v_i) = h_j\}$, where h_m is the lowest split point.

Lemma 4.3. *The time to explore a subtree of height h_m with $Nh_m + 1$ robots using* NCOCTA *is $\frac{N^{h_m}-1}{N-1} + h_m$.*

Proof. In perfect N-ary trees, the robots split equally between the N subtrees, so in every subtree there are as many robots as the height of the subtree (h_m). At every step of the algorithm, we visit a new node of the subtree (when a robot returns to his parent node, the robot that is located at the parent node can move to a different node). The number of nodes in the subtree is $\frac{N^{h_m}-1}{N-1}$ and the time to return to the root of the split node is h_m. The robots move simultaneously along the N subtrees, so the coverage time is: $\frac{N^{h_m}-1}{N-1} + h_m$ (for the 2-ary case the coverage time is $2^{h_m} - 1 + h_m$ with $2h_m + 1$ robots). See illustration in Figure 3. □

We denote the time of covering a subtree of height h_m as E_{h_m}. The time needed to explore a subtree with height h_{m-1} is the time needed to explore a subtree of height h_m multiplied by the number of subtrees with height h_m, summing with the runtime of DFS for a tree of height $h_{m-1} - h_m$:

$$E_{h_{m-1}} = 2^{h_{m-1}-h_m-1} \cdot E_{h_m} + 2^{h_{m-1}-h_m+1} - 2$$

This continues recursively up to h_1. Hence, the coverage time of a subtree with height h_1 is

$$E_{h_1} = 2^{h_1-h_2-1} \cdot E_{h_2} + 2^{h_1-h_2+1} - 2$$

Simple algebraic manipulations yields the following:

$$E_{h_1} = \{2^{h_1-h_2-1} \cdot \{2^{h_2-h_3-1}\{\dots\{2^{h_{m-2}-h_{m-1}-1} \cdot \{2^{h_{m-1}-h_m-1} \cdot (E_{h_m}) + 4) + 2\}...\} + 2\} + 2\} - 2$$

Recall that we initially assumed a given set of m splits $\{h_i\}_{i=1}^m$; we now turn to calculate the best choice for these values. By the last equation, increasing h_m will decrease the coverage time more significantly compared to increasing any h_i, $1 \leq i \leq m-1$. Therefore, we find the speedup factor under the assumption that $h_i = h_{i+1} + 1$. The time to explore a binary tree of height h_1 can now be calculated as:

$$2^{h_m} - 1 + h_m + 2(m-1),$$

and for the general case:

$$\frac{N^{h_m} - 1}{N-1} + h_m + 2(m-1).$$

Similar to the calculation of coverage time of a tree of height h_1, the time it takes to explore a tree with height H is the time it takes to explore a subtree of height h_1 multiplied by the number of subtrees of height h_1, summing with the runtime DFS for a tree of height $H - h_1$ as before. Hence, the overall coverage time for a full binary tree is

$$\begin{aligned} E_H &= 2^{H-h_1} E_{h_1} + 2(2^{H-h_1+1} - 1) - 2 \\ &= 2^{H-h_1}(2^{h_m} - 1 + h_m + 2(m-1) + 4) - 4, \end{aligned}$$

and for the general case:

$$\begin{aligned} E_H &= N^{H-h_1} E_{h_1} + 2\left(\frac{N^{H-h_1+1} - 1}{N-1}\right) - 2 \\ &= N^{H-h_1} \cdot \left(\frac{N^{h_m} + 2N - 1}{N-1} + h_m + 2(m-1)\right) - \frac{2N}{N-1} \end{aligned}$$

This yields the following.

Lemma 4.4. [3] $\mathsf{SF} > \dfrac{2N^{h_m+m}}{N^{h_m} + 1 + (N-1) \cdot (h_m + 2m)}$

Note that this implies that in order to maximize SF, it is desired to use the largest feasible value for m (as m appears in the exponent in the numerator and as a linear term in the denominator).

Lemma 4.5. $\mathsf{SF}(\mathsf{NCOCTA}) > 2 \cdot N^m - 1$ *for* $h_m > m + \log_N(m) + 4$

Note that the number of robots we need in order to obtain the speedup factor is $1 + Nh_1 + \sum_{i=2}^{m}(N^i - N^{i-1})h_i = 1 + Nh_1 + \sum_{i=2}^{m}(N^{i-1}(N-1))h_i$ in order to leave a trail of robots from the root to the leaves for every split. For $h_i = h_{i+1} + 1 = h_m + m - i \; \forall 1 \geq i \geq m$, after algebraic manipulation we obtain that the number of robots we need must be at least

$$k \geq 1 + N^m \cdot h_m + \frac{N^m - N}{N-1}$$

or, after some simplification, and denoting $\epsilon = 1/(N-1)$ (where $0 < \epsilon \leq 1$),

$$k \geq N^m \cdot (h_m + \epsilon) - \epsilon \,. \tag{1}$$

[3] Pure mathematical proofs are omitted due to space constraints.

It remains to select m and h_m so as to maximize SF. Taking $h_m = m + \log_N(m) + 5$, we get from Lemma 4.5 that $\mathsf{SF}(\mathsf{NCOCTA}) > 2 \cdot N^m - 1$, so we need to select the largest m that satisfies Eq. (1) with this h_m, namely,

$$k \geq N^m \cdot (m + \log_N m + 5 + \epsilon) - \epsilon \tag{2}$$

The optimal value turns out to be $m = \log_N k - \log_N \log_N k - 1$ (i.e., using m with the above h_m satisfies Eq. (2), while if we take $m-1$, Eq. (2) no longer holds).

With this choice of m and Lemma 4.5, we get:

Corollary 4.5.1. *Using* $m = \log_N k - \log_N \log_N k - 1$ *and* $h_m = m + \log_N(m) + 5$*, we get that* $\mathsf{SF} = \Omega\left(\frac{k}{\log_N k}\right)$.

Lemma 4.6. [3] $\mathsf{SF}(\mathsf{NCOCTA}) < 2 \cdot N^m$

For large H and k, it is possible to obtain a better asymptotic bound than that of Cor. 4.5.1.

Lemma 4.7. *Using* $h_m = \log_N \log_N k - \epsilon$ *and* $m = \log_N k - \log_N(h_m + \epsilon)$*, we get that* $\mathsf{SF} = \Omega\left(\frac{k}{\log_N \log_N k}\right)$.

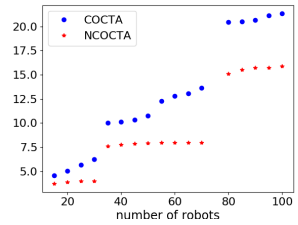
To better understand these equations, we bring the following example: the speedup factor on a 3-ary tree of height 7 using 16 robots is 5 ($m = \arg\max_m\{1+3^m(\log(m)+m+4)+0.5(3^m-3) \leq 16\} \Rightarrow m = 1$ hence $\mathsf{SF} = 2 \cdot 3^m - 1 = 5$). The speedup remains less than 6 ($2 \cdot 3^m$) until the number of robots is 58, when it becomes 17. This example demonstrates the inability of NCOCTA to exploit additional robots until there are enough robots to make an additional split. We address this problem next, presenting the COCTA algorithm.

4.3 COCTA Algorithm

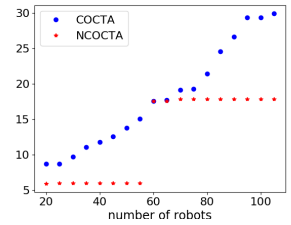
In order to increase the efficiency of our coverage algorithm, we would like to allow robots to start moving to adjacent subtrees before the root of the tree has become *finished*. We therefore introduce the *dripping-based* heuristic algorithm COCTA, which is similar to the NCOCTA algorithm, except that when T_v is inhabited, the robots always move to the parent node, leaving one robot behind (if v is the root, then the robots stop). Another enhancement in the algorithm is that when a node is not finished, we restrict the number of robots that move to the child, allowing a number of robots that does not exceed the number of nodes in the subtree, excluding the nodes that are finished. The rest of the robots can drip to a different subtree. The algorithm is shown in Alg. 2. We find the number of splits in the same way as before, and use the minimal h_m that gives us the proven speedup factor. We use the additional robots to disperse and explore a different subtree. Trivially, the speedup factor guaranteed by COCTA is not worse than NCOCTA. Going back to the example presented in section 4.2, the speedup on a perfect 3-ary tree was flat until there were enough robots to perform another split. Using COCTA the speedup of four additional robots increases to 8.6 (and up to 5 using 16 robots on NCOCTA).

4.4 Empirical Evaluation: COCTA vs. NCOCTA

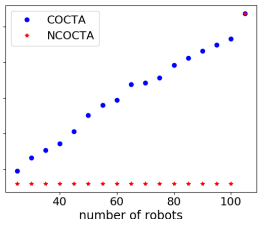
We have fully implemented both the COCTA and NCOCTA algorithms, and compared the coverage time of both algorithms with different perfect N-ary trees. The impact of the



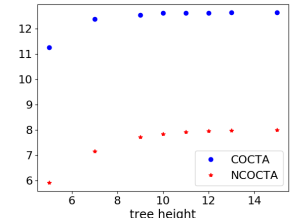
(a) SF vs number of robots on a perfect 2-ary tree (H=15)



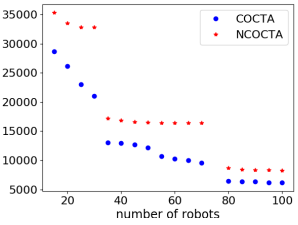
(b) SF vs number of robots on a perfect 3-ary tree (H=8)



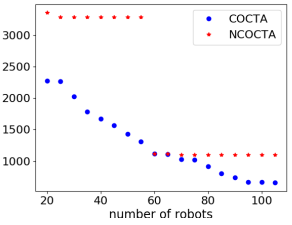
(c) SF vs number of robots on a perfect 4-ary tree (H=8)



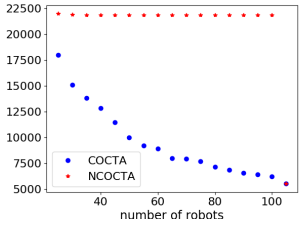
(d) SF on a perfect 2-ary tree using 60 robots with different tree heights



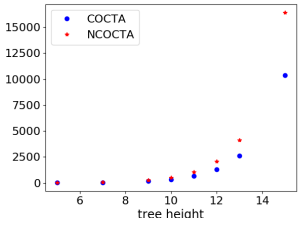
(e) Coverage time vs number of robots on a perfect 2-ary tree (H=15)



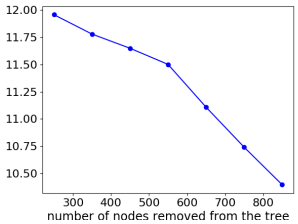
(f) Coverage time vs number of robots on a perfect 3-ary tree (H=8)



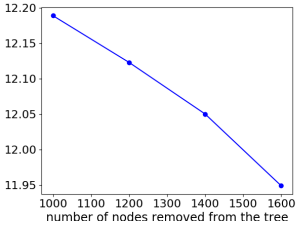
(g) Coverage time vs number of robots on a perfect 4-ary tree (H=8)



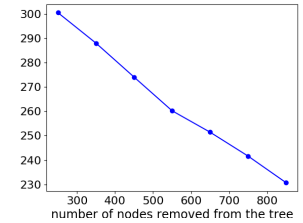
(h) Coverage time on a perfect 2-ary tree using 60 robots with different tree heights



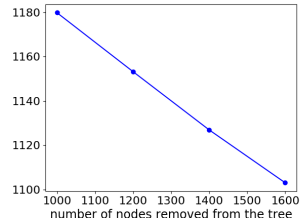
(i) SF($COCTA$) on general 2-ary tree (H=10) using 60 robots



(j) SF($COCTA$) on general 2-ary tree (H=12) using 60 robots



(k) COCTA coverage time on general 2-ary tree (H=10) using 60 robots



(l) COCTA coverage time on general 2-ary tree (H=12) using 60 robots

Figure 4: Experiment results.

number of robots on the coverage time is presented in Figures 4e, 4f, 4g. The influence of the number of robots on the speedup factor is presented in Figures 4a, 4b, 4c. As demonstrated in the example in section 4.2, NCOCTA's speedup does not increase much after reaching the number of robots allowed for a split, until an additional split is possible. This is indicated by the flat lines in Figures 4a, 4b, 4c. In contrast, COCTA is able to exploit these robots, as indicated by the associated graphs with an almost linear decrease in the coverage time. One can see from the figures that for the same number of robots, as the branching factor N grows, the coverage time becomes significantly smaller (and the speedup factor increases). The reason lies in the fact that when N is small, for example in binary trees, the robots do not have many options to disperse in order to explore the graph, thus they do not contribute to the coverage time compared to 4-ary trees, where it is easier to distribute the effort between the robots.

Additionally, we fixed the number of robots to examine how the tree's height impacts the coverage time. We used this model on a perfect 2-ary tree using 60 robots. This illustrates another contribution of the dripping-based heuristic algorithm COCTA: the deviation between the coverage times is significantly larger as the height ascends, as shown in Figure 4h. We can see in Figure 4d that there is no improvement in the speedup factor, as expected, since the number of robots determines the number of splits.

We have evaluated COCTA on general N-ary trees (not perfect) of heights 10 and 12 (Figures 4k, 4i and Figures 4l, 4j respectively). In order to create these imperfect trees, we defined a number of nodes to remove from the tree, and removed them from a predefined height (and all its subtrees) at random. We ran the simulation 10 times for 7 different heights, (hence, 70 times for every number of nodes to remove) and present the average coverage time and average speedup. This simulation mainly shows that COCTA works efficiently on general trees, even without the assumption of perfection. We can also see the comparison to a perfect tree (0 nodes removed) in this simulation and see a linear improvement in the coverage time.

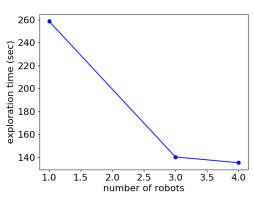
Algorithm 2 COCTA

while $root$ is not finished **do**
  for all $v \in V$ in which robots are located on (go over from the lowest to the highest height) **do**
    if v is finished **then**
      * *There are no robots in T_v subtrees, so we can return to the parent node* *\
      if $v \neq root$ **then**
        All robots from v go to the parent of v.
      else
        All robots from v stop.
      end if
    else if $\exists$ u a child of v such that u is unfinished **then**
      * *v is unfinished, hence, there are still nodes to explore. We can move to a child If there are more robots than nodes to explore, we can use the rest of the robots to explore a different subtree**\
      if $\exists h_i$ such that the $h(v) = h_i$ **then**
        Leave one robot at v and split the rest equally among the children.
        If there are more robots than nodes to explore, move the rest of the robots to the parent
      else if $h(v) \leq h_1$ **then**
        Select a child u of v such that u is unfinished.
        Move all the robots in v to u leaving one robot in v.
        If there are more robots than nodes to explore, select another child to move the rest to.
      else
        Select a child u of v such that u is unfinished.
        Move all the robots in v to u.
        If there are more robots than nodes to explore, leave one robot at the node and select another child to move the rest to.
      end if
    else
      * *v is inhabited, the subtree is explored but there are still robots in the subtree. The robots leave one robot behind to maintain the communication and drip to the parent node.* *\
      All robots from v go to the parent of v leaving one robot in v.
    end if
  end for
end while

5 Simulated and Real Deployment in an Office Building

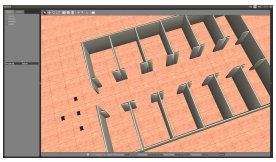
We simulated the coverage in a realistic simulation (ROS/Gazebo) for exploring an office building using the COCTA algorithm, as presented in Figure 5. We compared the results of the coverage time using one, three and four robots. From the results that are presented in Figure 5a it is clear that by using three robots, the coverage time decreases from 260 seconds to 140 seconds, i.e., close to the theoretically proven speedup factor. An additional robot does not significantly improve the coverage time, since the tree being modeled was narrow and high with a branching factor of 3. We also implemented the coverage algorithm (COCTA) on real robots, the Hamster robots [4], as seen in Figure 5b. The

[4] http://www.cogniteam.com/hamster4.html



(a) Coverage using 1,3,4 robots

(b) Hamster robots



(c) Office building map on ROS/Gazebo

Figure 5: Realistic Simulation

robots explored an office building with a similar tree representation to the one used in the ROS/Gazebo simulation in order to compare the two results.

The tree that modeled the office building is a general 3-ary tree of height 7. The Hamster robots moved fast (0.5 meters/sec). When using 3 robots for the coverage, the speedup factor of the (theoretical) tree environment was 2.05. The speedup in the ROS/Gazebo simulation was 1.843, and on the real robots the speedup factor was only 1.4816. We believe that the smaller speedup factor values in both the ROS/Gazebo simulation and on the actual robots is due to time spent on coordination and synchronization between the robots. Regardless, the ability of the Hamster robots to efficiently explore an office building while maintaining communication is very promising for real applications such as search and rescue.

6 Conclusion and Future Work

In this paper we developed a novel algorithm for exploring a perfect N-ary tree. We provide a theoretical analysis of the coverage time using the notion of a speedup factor. We improved the theoretically-proven NCOCTA algorithm by using a dripping-heuristic algorithm, COCTA, which is shown in extensive simulations to significantly decrease the coverage time of NCOCTA. Additionally, we have implemented our solutions on a realistic robotic simulation and deployed our solution on real robots, which demonstrates in practice the efficiency of our coverage algorithms in a real office building setting.

In the future we plan to extend our solution to general trees both in theory and in practice. Additionally, we would like to find theoretical tight bounds on possible speedup factors (lower and/or upper bounds).

References

[Banfi *et al.*, 2016] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5491–5498, 2016.

[Brass *et al.*, 2011] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[Cabrera-Mora and Xiao, 2012] Flavio Cabrera-Mora and Jizhong Xiao. A flooding algorithm for multirobot exploration. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):850–863, 2012.

[Förster and Wattenhofer, 2016] Klaus-Tycho Förster and Roger Wattenhofer. Lower and upper competitive bounds for online directed graph exploration. *Theoretical Computer Science*, 655:15–29, 2016.

[Fraigniaud *et al.*, 2004] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R Kowalski, and Andrzej Pelc. Collective tree exploration. In *Proceedings of Latin American Symposium on Theoretical Informatics*, pages 141–151, 2004.

[Gabriely and Rimon, 2001] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):77–98, 2001.

[Hazon and Kaminka, 2008] Noam Hazon and Gal Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114, 2008.

[Jensen and Gini, 2013] Elizabeth A Jensen and Maria L Gini. Rolling dispersion for robot teams. In *Proc. 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2013.

[Jensen *et al.*, 2014] Elizabeth A Jensen, Ernesto Nunes, and Maria Gini. Communication-restricted exploration for robot teams. In *Proc. AAAI Workshops on Artificial Intelligence*, 2014.

[Pei and Mutka, 2012] Yuanteng Pei and Matt W Mutka. Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1551–1556. IEEE, 2012.

[Rogge and Aeyels, 2007a] Jonathan Rogge and Dirk Aeyels. A novel strategy for exploration with multiple robots. In *Proc. 4th Int. Conf. on Informatics in Control, Automation and Robotics*, pages 76–83, 2007.

[Rogge and Aeyels, 2007b] Jonathan Rogge and Dirk Aeyels. Sensor coverage with a multi-robot system. In *Proc. IEEE 22nd Int. Symp. on Intelligent Control*, pages 71–76, 2007.

[Wilkinson and Allen, 1999] Barry Wilkinson and Michael Allen. *Parallel Programing*, volume 999. Prentice hall Upper Saddle River, NJ, 1999.