

Optimal Constrained Graph Exploration*

CHRISTIAN A. DUNCAN

Department of Computer Science
University of Miami
duncan@cs.miami.edu

STEPHEN G. KOBOUROV[†]

Department of Computer Science
University of Arizona
kobourov@cs.arizona.edu

V. S. ANIL KUMAR

Basic and Applied Simulation Science (CCS-5)
Los Alamos National Laboratory
anil@lanl.gov

Abstract

We address the problem of constrained exploration of an unknown graph $G = (V, E)$ from a given start node s with either a tethered robot or a robot with a fuel tank of limited capacity, the former being a tighter constraint. In both variations of the problem, the robot can only move along the edges of the graph, i.e. it cannot jump between non-adjacent vertices. In the tethered robot case, if the tether (rope) has length l , then the robot must remain within distance l from the start node s . In the second variation, a fuel tank of limited capacity forces the robot to return to s after traversing C edges. The efficiency of algorithms for both variations of the problem is measured by the number of edges traversed during the exploration. We present an algorithm for a tethered robot which explores the graph in $\Theta(|E|)$ edge traversals. The problem of exploration using a robot with a limited fuel tank capacity can be solved with a simple reduction from the tethered robot case and also yields a $\Theta(|E|)$ algorithm. This improves on the previous best known bound of $O(|E| + |V| \log^2 |V|)$ in [4]. Since the lower bound for the graph exploration problems is $|E|$, our algorithm is optimal within a constant factor, thus answering the open problem of Awerbuch, Betke, Rivest, and Singh [3].

1 Introduction

Robot navigation is often modeled as the problem of exploring an unknown graph, $G = (V, E)$. The robot explores the graph by starting at a fixed start node s and performing edge traversals. As the robot explores

G it constructs a map of the known portions of the graph. That is, once the robot has visited a node or traversed an edge, it can recognize them if it encounters them again; this can be done by marking the nodes [15] or using pebbles [6]. The graph $G = (V, E)$ is explored once all the vertices in V have been visited and all the edges in E have been traversed; we will assume that the graph to be explored is connected. Let $\ell(e)$ denote the length of edge e - we will say that the time needed to explore e equals $\ell(e)$ (i.e., assuming unit speed). If $\ell(e) = 1$ for all the edges, we have the unweighted case, where a trivial lower bound for the running time of an exploration algorithm measured by the number of edge traversals is $|E| + |V|$; the lower bound in the weighted case is $\sum_e \ell(e)$. Almost all the results on graph exploration only pertain to unweighted graphs. For the unconstrained exploration problem the classical search algorithms, breadth-first search (BFS) and depth-first search (DFS), e.g. see [12], are optimal within a constant factor.

The piecemeal (or fuel constrained) search model, introduced by Betke, Rivest and Singh [9], adds two natural constraints to the graph exploration problem:

- *Continuity*: the robot must traverse a continuous path in the graph, i.e., it cannot jump from one point to another (unlike in [1], who allow moves of this sort). This constraint is relevant in problems where the traversal corresponds to physical exploration of terrain.
- *Interruptibility*: the robot must return to the start node after traversing C steps. C is typically at least $2(1 + \alpha)r$, where r is the distance to the farthest node from the start node s and α is any nonnegative constant.

In this paper we consider another seemingly even

*Some of this work was done while the authors were at the Max-Planck-Institut für Informatik.

[†]This research partially supported by NSF under Grant CCR-9625289.

more restricted problem of graph exploration: exploration by a tethered robot with a rope of length l . In this model, the robot is tied to the start node by a rope and is forced to match every forward traversal of an edge with a backward traversal, rewinding the rope, in a first-in last-out stack order. In practical terms, the rope can be a fuel line, or a communication line, or a safety line.

Although the tethered robot is not constrained to return to s periodically, it might be forced to backtrack a great deal even to visit an adjacent vertex. However, the two models - the fuel constrained and tethered search - are equivalent within constant factors, as shown in Section 2, i.e., for appropriate rope and fuel lengths, the exploration times are equivalent within constant factors.

Note that neither the fuel constrained search problem nor the tethered robot search problem can be solved using the classical search algorithms. BFS violates the continuity constraint and DFS violates either the interruptibility constraint or the maximum rope length, as shown in the examples of [4]. We show how to solve both the piecemeal search and the tethered robot search problems by a conceptually simple Closest First Exploration (CFX) algorithm.

1.1 Previous Work

Exploration and navigation in unknown terrains is a well studied problem (see the survey of Rao *et al* [19]). Exploration in geometric settings has been studied by Blum, Raghavan, and Schieber [11], Bar-Eli, Berman, Fiat, and Yan [5], Deng and Papadimitriou [13] and Hoffmann *et al* [16]. Papadimitriou and Yanakakis [18] consider finding shortest paths in various unknown graphs and Blum and Chalasani [10] give a “k-trip” shortest path algorithm.

Rivest and Schapire [20] study a finite state model for exploration, while Deng and Papadimitriou [14] and Albers and Henzinger [1] model it on a directed graph. Bender and Slonim [7] consider the problem with two cooperating robots. Randomized robot navigation was addressed by Berman *et al* [8].

In situations corresponding to physical settings the exploration is modeled by an undirected graph. Without the two piecemeal constraints (continuity and interruptibility) an undirected graph can be explored in $\Theta(|E| + |V|)$ using either BFS or DFS. Panaite and Pelc [17] address the problem of minimizing the constants in such linear time exploration, achieving a bound of $|E| + \Theta(|V|)$.

The problem of piecemeal (or fuel constrained) exploration of an undirected graph was introduced by Betke, Rivest, and Singh [9] where an $O(|E| + |V|)$ algorithm is presented for two classes of graphs: city-

block graphs and grid graphs with rectangular obstacles. Awerbuch, Betke, Rivest, and Singh [3] present an $O(|E| + |V|^{1+\epsilon})$ algorithm for general graphs and pose the open problem of finding a linear time algorithm for general graphs. Awerbuch and Kobourov [4] employ sparse neighborhood covers [2] in an $O(|E| + |V| \log^2 |V|)$ algorithm, which is the best known result thus far.

The problem of finding a linear time algorithm has been open not only for general graphs but even for special classes of graphs such as planar graphs and grid graphs with non-convex obstacles.

1.2 Our Results

We settle the open problem in Awerbuch, Betke, Rivest, and Singh [3] by showing an algorithm for fuel constrained and tethered robot exploration that runs in $O(|E|)$ steps. We show that G can be explored by a tethered robot in $\Theta(|E| + |V|/\alpha)$ steps using a rope of length $(1 + \alpha)r$, where r is the *radius*, i.e., the distance to the farthest node from the start node s ; from our reduction described in Section 2, this immediately implies an $O(|E|(1 + \beta)/(\alpha(\beta - \alpha)))$ bound for fuel constrained exploration with a fuel tank of size $2(1 + \beta)r$, $\beta > \alpha$.

We also consider the weighted case, where edge lengths can be arbitrary, and show that the number of steps is still $O(\ell(E) = \sum_{e \in E} \ell(e))$, but with a rope of length $(1 + \alpha)r + \ell_{max}$, where $\ell_{max} = \max_e \{\ell(e)\}$. This is the first such result on graph exploration which holds for weighted graphs.

We also consider the case where the radius is not known a priori, and has to be deduced incrementally. We show that our algorithm can be extended to work in $\Theta(|E| + |V|/\alpha)$ and $\Theta(|E|/\alpha)$ time for tethered and fuel constrained explorations, respectively. This leads to a solution for the *treasure hunting* problem defined in [3], which requires searching for a treasure node t in an unknown infinite graph. Our algorithm solves this problem in time $O(|G'|)$, where G' is the subgraph induced by vertices within a distance of $(1 + \alpha)d(s, t)$ from the start node, $d(s, t)$ being the distance of the treasure from s .

Organization We describe the notation and the relationship between the two exploration models in Section 2. In Section 3.2 we present the CFX algorithm which solves the tethered graph exploration problem when all edge lengths are 1 (i.e., the unweighted case) and analyze its running time. In Section 4, we extend this algorithm to the case where edge lengths can be arbitrary. In Section 5 we present and analyze a simple modification for cases when the radius is not known in advance. We also extend this solution to solve the treasure hunting problem. Section 6 concludes with a few open problems.

2 Preliminaries

Graph $G(V, E)$ will denote the graph to be explored from s , the start node. Each edge e has length $\ell(e)$; in the unweighted case, $\ell(e) = 1$ for each edge. We will also denote ℓ_{max} to be the length of the longest edge, i.e., $\ell_{max} = \max_e \{\ell(e)\}$. Let r be the *radius* of the graph, equal to the distance to the farthest vertex from s . For now we assume that r is known, or at least is bounded above by a known value.

Simple Lower Bounds For the tethered robot exploration, the rope must have length at least $r + \ell_{max}$, else the graph cannot be fully explored. An instance where this is necessary is a triangle with nodes $\{s, a, b\}$ and edges $\{(s, a), (s, b), (s, c)\}$. Let each of the edges have length ℓ . Then the $r = \ell_{max} = \ell$, but to explore all the three edges, the rope length has to be $2\ell = r + \ell_{max}$. Similarly, for piecemeal exploration in this instance, the fuel tank must have size at least $3\ell = 2r + \ell_{max}$.

In our algorithms, we will actually require a slightly longer rope or larger fuel tank size- for the tethered case, we will assume a rope of length $(1 + \alpha)r$, and for the piecemeal case, we will assume a tank of size $2(1 + \beta)r$. As we show below, the two models are related within constant factors, and so we will only describe algorithms for the tethered case.

LEMMA 2.1. *The fuel constrained exploration problem with a tank of size $2(1 + \beta)r$ can be reduced to the tethered exploration problem with a rope of length $(1 + \alpha)r$, for any $\beta > \ell_{max}/r + \alpha$; the running time increases by a factor of $\frac{1 + \beta}{\beta - \alpha}$. Conversely, the tethered exploration problem with a rope of length $2(1 + \beta)r$ can be reduced to the fuel constrained exploration problem with a rope of length $(1 + \beta)r$, in the same time.*

Proof. We first describe the reduction from piecemeal search to a tethered search. Let A be an algorithm that performs a tethered robot exploration, with a rope of length $(1 + \alpha)r$, in time T . An algorithm B for fuel constrained exploration with a tank of size $2(1 + \beta)r$, with $\beta > \ell_{max}/r + \alpha$ can be obtained in the following manner (as in [3] for the unweighted case). Consider time steps $t_0 = 0, t_1, t_2, \dots$ in the exploration by algorithm A : for each $i > 0$, t_i is the first time instant after t_{i-1} such that the sum of the edge lengths explored after time t_{i-1} exceeds $(\beta - \alpha)r$; then, $t_i \leq t_{i-1} + (\beta - \alpha)r + \ell_{max} \leq 2(\beta - \alpha)r$. For each i , B emulates A from time step t_{i-1} to t_i , then goes to s for refueling, and comes back. The robot is always within distance $(1 + \alpha)r$ from s during the course of algorithm A . Therefore, the amount of fuel needed to go to s and back from any node v is at most $2(1 + \alpha)r$, and the total fuel used is $2(\beta - \alpha)r + 2(1 + \alpha)r = 2(1 + \beta)r$. Since there

is a refueling operation taking time $2(1 + \alpha)r$ after at least $(\beta - \alpha)r$ emulated steps of algorithm A , algorithm B takes time $T + 2T(1 + \alpha)/(\beta - \alpha) \leq 2T(1 + \beta)/(\beta - \alpha)$.

Similarly, algorithm B for fuel constrained exploration with a tank of $(1 + \beta')r$ can be transformed to an algorithm A for a tethered robot exploration by a rope of length $2(1 + \beta')r$ very easily. B returns to the start node within $(1 + \beta')r$ steps. The tethered robot needs to retrace its steps (to un-entangle itself), and a rope of length $2(1 + \beta')r$ is sufficient for this. The time taken by algorithm A is at most twice that of algorithm B .

For any graph $G(V, E)$ with edge lengths $\ell()$, let $P_G(v, w)$ be the shortest path between v and w in H . Let $d_G(v, w) = |P_G(v, w)|$ represent the distance between vertices v and w in G , i.e., $d_G(v, w) = \sum_{e \in P_G(v, w)} \ell(e)$. the number of *edges* in the path $P_G(v, w)$. We also expand our definition to include the distance between sets of vertices $U, W \subseteq V$. Let $P_G(U, W)$ be the shortest path in G between any $u \in U$ and $w \in W$. Similarly, we define $d_G(U, W) = \min_{u \in U, w \in W} d_G(u, w)$ to be the *minimum* distance between the two sets in G . In the same way, let $\Delta_G(U, W) = \max_{u \in U, w \in W} d_G(u, w)$ be the *maximum* distance. When a set consists of a single node, we will sometimes drop the set notation - so $d(s, U)$ will be used to denote $d(\{s\}, U)$, for a subset $U \subset V$.

3 Efficient Exploration of Unweighted Graphs

In this section, we will assume that the graph is unweighted, i.e., $\ell(e) = 1$ for each edge e . As before, s is the start node and r denotes the radius from s .

As the fueled robot is a less constrained version, we will focus on the tethered robot case, and assume that the robot has a rope of length $(1 + \alpha)r$ for some $\alpha > 0$.

3.1 Bounded Depth First Exploration

Before presenting the optimal solution for tethered robot exploration we briefly examine some related non-optimal solutions, one of which is necessary in the final algorithm. Let us assume now that the robot has a rope length $l = |V|$. For any graph $G = (V, E)$ notice that $r < |V|$. It is easy to explore a graph using a tethered robot with rope length $l = |V|$ using a DFS traversal. In fact, for a tethered robot with such rope length, DFS is optimal. The algorithm takes exactly $2|E|$ steps as every edge is traversed twice, and, since a tethered robot must backtrack over every step, this is optimal. If we allow the rope to be of infinite length, the problem reverts to the unconstrained robot exploration. Here, one may explore the graph optimally using $|E| + O(|V|)$ steps [17]. However, if the length of the rope is $O(r)$, there are simple cases in which the

```

bDFX( $v, l$ )
  if  $v$  is tagged
    return
  if  $l = 0$ 
    // Maximum length reached. Must backtrack
    if  $v$  has any unexplored edge
      Mark  $v$  as incomplete
    else
      Mark  $v$  as explored
    return
  Mark  $v$  as tagged
  For each unexplored edge  $(v, w) \in E$ 
    MOVE ROBOT to  $w$  along edge  $(v, w)$ 
    Mark  $(v, w)$  as explored
    Call bDFX( $w, l - 1$ )
    MOVE ROBOT BACK from  $w$  to  $v$  along edge  $(v, w)$ 
  Mark  $v$  as explored

```

Figure 1: The recursive bDFX algorithm from node v with a remaining rope length of l

standard DFS fails [4]. More specifically, the behavior of DFS is not well-defined when the robot reaches its maximum rope length. It certainly cannot continue further, but how does it proceed?

One obvious solution is to allow the robot to ignore any edges it cannot reach and proceed by backtracking, returning along the rope. Thus, the search has a bounded depth constraint associated with it. For clarity, we present this bounded depth first exploration (bDFX) algorithm in Figure 1.

We classify the edges of G into two types, *explored* and *unexplored*. Similarly, the vertices are of four types, *explored*, *unexplored*, *incomplete*, and *tagged*. Initially, every edge and vertex is unexplored. Incomplete vertices are those that have been partially explored, but have some incident unexplored edges. Thus, a vertex becomes explored only when all of its incident edges are explored. Tagged vertices are used by the bDFX algorithm to temporarily mark vertices on the search stack.

The recursive bDFX algorithm simply starts at a node v with a given rope length l and recursively visits all unvisited edges incident to v . We also have the terminal condition that no edges are visited if the length is 0. At the end of the call the robot is back to its original position v with the same initial rope length. From the algorithm, one can see that the amount of steps taken by the robot is exactly $2|E'|$ where E' is the set of edges marked *explored* by bDFX.

At first glance, this simple modification may appear to solve the problem of tethered exploration. Unfortunately, this is not the case, as shown in Figure ?? . After

termination, the algorithm may leave many edges *unexplored*.

3.2 Optimal Tethered Exploration

Notation At every stage, we also maintain the complete known subgraph $G^* \subseteq G$ consisting of all edges and vertices explored or incomplete so far. In our algorithm, we will maintain a collection of trees $\mathcal{T} = \{T_1, T_2, \dots\}$. In particular, we are interested in $d_{G^*}(s, T_i)$ for $T_i \in \mathcal{T}$. Let $s_i \in T_i$ be any vertex such that $d_{G^*}(s, s_i) = d_{G^*}(s, T_i)$. In other words, s_i is the closest known vertex in T_i to s . For a given tree T , we will also use $\Delta_T(v, T)$ to denote $\Delta_T(\{v\}, T)$.

We are also concerned with the sizes of different sets. Unless otherwise specified, for any graph H , let $|H|$ represent the number of vertices, as opposed to edges, in the graph. We note the exception that, for any path P , $|P|$ represents the number of edges.

Basic Idea In order to completely explore the graph, we enhance the bDFX algorithm to run in phases until all vertices and edges are explored. At each phase, we start with a set $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$ of vertex disjoint subtrees of G whose union contains all currently incomplete vertices. Initially, this set consists of one subtree containing only s . For each phase, a particular tree T_i is chosen from \mathcal{T} , and we start exploring from all the incomplete vertices in T_i . We will again construct trees on the region of the graph explored from the incomplete vertices in T_i - these trees will be used in subsequent iterations. For being able to explore a non-trivial part of the graph starting from the incomplete vertices of T_i , we require that $\Delta_{T_i}(s_i, T_i)$ be upper bounded. However, it will turn out that just having each T_i “small” is not sufficient- if T_i is too small, it would be hard to account for the cost of the trip from s to the root of the T_i , and doing the exploration from the incomplete vertices in T_i (in the case that not much was explored from the incomplete vertices in T_i , compared to the length of the path from s to T_i).

To take care of this problem, we introduce an additional constraint that each T_i be also “lower bounded”. We ensure this by taking each incomplete tree, and pruning tree T_i into smaller subtrees $T_{i_0}, T_{i_1}, \dots, T_{i_k}$, and the resulting subtrees replace T_i in \mathcal{T} ; the resulting trees maintain some minimum size and maximum depth. Then, for each incomplete vertex in T_{i_0} visited using a standard depth first traversal on the tree, we perform a bDFX from that vertex. This exploration in turn creates several newly explored subtrees T_{k+1}, T_{k+2}, \dots which are also added to \mathcal{T} .

For any tree T rooted at a vertex v , we have that $|T| > \Delta_T(v, T)$. After each phase, the *minimum-size*

criterion we maintain is that for all $T \in \mathcal{T}$, $|T| \geq \alpha r/4$.¹

Figure 2 describes the core elements of the CFX algorithm. The idea is to pick the tree T closest to the start node s and, after pruning it, run **bDFX** on all of the incomplete vertices in T . The total time taken by the robot is equivalent to the time spent getting from s to $s_i \in T$ and back plus the time to visit all incomplete vertices in T plus the time needed to perform the various iterations of **bDFX**. The only mysterious step lies in the pruning operation, explained intuitively earlier, and whose importance becomes apparent in the analysis. The process simply prunes the tree rooted at v so that the depth of the tree is small, i.e., $\Delta_T(v, T) \leq \alpha r/2$. However, to ensure that the pruned subtrees, T_w , have proper size, we do not prune a subtree with few vertices, i.e., we maintain that $\Delta_{T_w}(w, T_w) \geq \alpha r/4$.

3.3 Analysis

The proof for the running time and correctness of the CFX algorithm relies on maintaining the minimum-size criterion for all trees in \mathcal{T} . Our proofs begin with several smaller lemmas and build up to the main theorem of this section.

LEMMA 3.1. *Let \mathcal{T}' be the collection of trees formed in any run of **explore** on a tree T rooted at vertex s_i with rope length l . For any tree $T' \in \mathcal{T}'$, if T' is incomplete, then $|T'| > l - d_{G^*}(s, s_i) - \Delta_T(s_i, T)$. Furthermore, if $l - d_{G^*}(s, s_i) - \Delta_T(s_i, T) > 0$, then the initial tree T becomes explored.*

Proof. Following the procedure **explore**, let T' be an incomplete tree created after running **bDFX** from vertex v . Let $G' = (V', E')$ be the subgraph explored during the **bDFX** from v . We can bound the length of rope remaining when the robot is at vertex v by the current path taken from s

$$l' = l - d_{G^*}(s, s_i) - d_T(s_i, v) \geq l - d_{G^*}(s, s_i) - \Delta_T(s_i, T).$$

Let $w \in V'$ be an incomplete vertex. After the running of **bDFX** with a rope length l' , in order for w to remain incomplete, the remaining length, d , at vertex w must have been 0, see the code in Figure 1. Therefore, there must exist a vertex disjoint path of new explored edges from v to w whose length is l' . This implies that $|T'| > l'$.

To see the second part of the lemma, assume that after termination T remains incomplete. Let v be any incomplete vertex in T . During the depth first traversal of T , vertex v must have been visited. Since l' is the length of rope remaining, we know that $l' =$

$l - d_{G^*}(s, s_i) - d_T(s_i, v) > 0$. After calling the function **bDFX** with a positive rope length, the initial vertex v can only be marked *explored*. This contradicts the assumption that v was incomplete after termination. Therefore, T cannot be incomplete.

LEMMA 3.2. *After every iteration, any incomplete vertex in G^* belongs to some tree $T \in \mathcal{T}$.*

Proof. Vertices are marked *incomplete* only in the **bDFX** function. After each call, these vertices, by definition, belong to the graph G' and the corresponding spanning forest \mathcal{T}' which is added to \mathcal{T} during **explore**. Let us look at the other possible changes to \mathcal{T} . The **prune** function simply breaks trees up, not removing any vertices. After each iteration in the main loop of CFX, only explored trees are removed. The final merging step in the loop simply joins trees together, also not eliminating any vertices. Since initially the only incomplete vertex, s , belongs in \mathcal{T} , after each iteration, every incomplete vertex in G^* must still lie in some tree in \mathcal{T} .

LEMMA 3.3. *If $G^* \neq G$, there exists an incomplete vertex v such that $d_{G^*}(s, v) \leq r$.*

Proof. Assume not. Let v be the closest incomplete vertex to s . Then, $d_{G^*}(s, v) > r$. Let $P = P_G(s, v)$ be the true shortest path from s to v . Notice that $|P| \leq r$ by the definition of the radius. Let $e = (u, w)$ be the first edge on P not in G^* . This must exist otherwise $d_{G^*}(s, v) = |P| \leq r$. This implies that vertex u is also incomplete in G^* . However, $d_{G^*}(s, u) = d_G(s, u) \leq r < d_{G^*}(s, v)$ contradicts the fact that v is the closest incomplete vertex.

LEMMA 3.4. *After pruning a tree T rooted at v , the maximum depth of T is less than or equal to $\max D$. In other words, $\Delta_T(v, T) \leq \max D$. If $|T| \geq \min D$, then after pruning, $|T| \geq \min D$. For any subtree T_w created after pruning, $|T_w| > \max D - \min D$.*

Proof. Assume the first part of the lemma is not true. After the termination of function **prune**, let $u \in T$ be any vertex such that $d_T(v, u) > \max D$. Let w be the ancestor of u such that $d_T(v, w) = \min D$. We now have $\Delta_T(v, T_w) > \max D$. By the condition in function **prune**, T_w should be separated from T and thus $u \notin T$.

For the second part, let T be the original tree and T' be the resulting pruned version of T . Assume the contrary that $|T'| < \min D$. Let $w \in T$ be the root of any pruned subtree, T_w . Since w was chosen to be pruned, it necessarily follows that $d_T(v, w) = \min D$. Let u be the parent of w in T . Let $P = P_T(v, u)$. Since u

¹For convenience, we assume that $\alpha r/4$ is an integer.

cannot be pruned, $P \subseteq T'$. This, however, implies that $|T'| \geq |P| + 1 = \text{minD}$.

For the final part, let T_w be any pruned subtree of T . Recall that $d_T(v, w) = \text{minD}$. By the condition in function `prune`, we know that $\Delta_T(v, T_w) > \text{maxD}$. Therefore, we know that $\Delta_{T_w}(w, T_w) = \Delta_T(w, T_w) = \Delta_T(v, T_w) - d_T(v, w) > \text{maxD} - \text{minD}$. This completes the proof as $|T_w| > \Delta_{T_w}(w, T_w)$.

THEOREM 3.1. *Any unknown graph $G = (V, E)$ with source node s and radius r can be explored by a tethered robot with rope length $(1 + \alpha)r$ for $0 < \alpha < 1$ in $\Theta(|E| + |V|/\alpha)$ edge traversals.*

Proof. In the first step, when the only tree consists of $\{s\}$, a `bDFX` is done from s , and the total cost is at most $2|E|$.

We first show that the minimum size criterion is maintained after each subsequent step, i.e., for every tree $T \in \mathcal{T}$, $|T| \geq \alpha r/4$. After the initial iteration of the loop, one can see that this criterion holds. In fact, after the first loop, if the graph is still incomplete, \mathcal{T} contains only one tree T and, by Lemma 3.1, $|T| \geq (1 + \alpha)r > \alpha r/4$.

During every successive iteration of the main loop, we choose the tree $T_i \in \mathcal{T}$ closest to s with the closest vertex $s_i \in T_i$. Let T be the resulting pruned tree of T_i containing s_i . Since $\text{minD} = \alpha r/4$, Lemma 3.4 shows that T still maintains the minimum size criterion. Furthermore, Lemma 3.4 also shows that for each pruned subtree T_w , $|T_w| > \text{maxD} - \text{minD} = \alpha r/4$. Since the merging step only increases the size of trees and all explored trees are removed, we only need to show that for every incomplete tree T' created in `explore` $|T'| \geq \alpha r/4$.

From Lemma 3.3, we see that $d_{G^*}(s, s_i) \leq r$. Lemma 3.4 states that $\Delta_T(s_i, T) \leq \text{maxD} = \alpha r/2$. Let v be any vertex in T which was incomplete prior to calling `explore` on T and let T' be the newly explored tree discovered by the `bDFX` function on v . From the `explore` procedure, we know that $l' = l - d_{G^*}(s, s_i) - d_T(s_i, v) \geq (1 + \alpha)r - r - \text{maxD} = \alpha r/2 > 0$. If T' is incomplete, from Lemma 3.1, we see that $|T'| > l' \geq \alpha r/2$.

Therefore, after each iteration of the main loop we see that for any tree $T \in \mathcal{T}$, $|T| \geq \alpha r/4$. In addition from Lemma 3.1, we know that, after the `explore` function on a tree T terminates, T becomes explored and is removed from \mathcal{T} .

Since all steps taken by the robot are performed in the `explore` and `bDFX` functions, we can analyze the exploration time of the algorithm by analyzing these two procedures. The first step of the algorithm, when the only tree is $\{s\}$, incurs of cost of at most $2|E|$, and we now analyze the remaining cost. Let us examine

again the pruned tree T given to the `explore` function. Let us break the exploration steps into three groups, $S_1(T), S_2(T), S_3(T)$. Let $S_1(T) = 2d_{G^*}(s, s_i)$ be the number of steps taken by the first and last lines of the algorithm performing a roundtrip from s to s_i . Let $S_2(T) = 2|T|$ be the number of steps taken to perform a depth first traversal of T . Let $S_3(T)$ be the number of steps taken by all calls to `bDFX` for one call of `explore`.

Lemma 3.3 implies that $d_{G^*}(s, s_i) \leq r$. Since $|T| \geq \alpha r/4$, we can bound $S_1(T) \leq r \leq 8|T|/\alpha$. The traversal time taken by the `explore` function is $S_1(T) + S_2(T) + S_3(T) \leq (2 + 8/\alpha)|T| + S_3(T)$.

Let \mathcal{T}_e be the set of all trees explored. Since `bDFX` only traverses unexplored edges, we immediately see that $\sum_{T \in \mathcal{T}_e} S_3(T) = 2|E|$. We now wish to bound the size of all trees explored. After any iteration of the algorithm, all trees in \mathcal{T} are vertex disjoint. One would then assume that the size of all trees is bounded by $|V|$. However, there is a little overlap between successive iterations. If we let T be the current tree explored, any incomplete vertex $v \in T$ becomes complete after the `explore` function. Some of such vertices v might become the roots of incomplete trees in future phases, though the tree T is removed from \mathcal{T} subsequently; each such vertex can be part of at most two trees. We can then bound the sum by $\sum_{T \in \mathcal{T}_e} |T| \leq 2|V|$. Therefore, the total steps \mathcal{S} taken by the `CFX` algorithm are

$$\begin{aligned} \mathcal{S} &= \sum_{T \in \mathcal{T}_e} (S_1(T) + S_2(T) + S_3(T)) \\ &\leq 2|E| + \sum_{T \in \mathcal{T}_e} (2 + 8/\alpha)|T| \\ &\leq 2|E| + (2 + 8/\alpha)2|V|. \end{aligned}$$

By Lemma 3.2, any incomplete vertex in G^* must belong to some tree $T \in \mathcal{T}$. Since the algorithm terminates, $\mathcal{T} = \emptyset$ implies that there are no more incomplete vertices in G^* . Therefore, $G^* = G$ and the graph is fully explored in $\Theta(|E| + |V|/\alpha)$ time.

4 Extension to weighted graphs

We now assume that the graph $G(V, E)$ is weighted. Each edge $e \in E$ has a length $\ell(e)$, and the time taken to traverse e is proportional to $\ell(e)$. The notion of $d()$ defined earlier in Section 2 is now extended to be the shortest distance with respect to these lengths, and r is defined as $\max_u \{d(s, u)\}$. Also, in this section, we will assume that the rope has length $(1 + \alpha)r + 2\ell_{max}$; from the discussion in Section 2 $r + \ell_{max}$ is necessary.

We will basically modify the `bDFX` and `CFX` algorithms to take care of the edge lengths in the following manner:

1. The `weighted-bDFX` algorithm described in Figure 3 below, is obtained by modifying the `bDFX` algorithm in the following natural manner: (i) on

traversing an edge e , the length of the remaining rope decreases by $\ell(e)$, instead of 1, and (ii) the robot has to backtrack from a node if the length of the remaining rope is less than the lengths of all the unexplored edges incident on that vertex - even if there is a positive rope length that is remaining.

2. The weighted version of the CFX algorithm is called **weightedCFX**, and has two differences from algorithm **CFX**. The first is that the call to **explore** is made with a rope length of $(1 + \alpha)r + 2\ell_{max}$, instead of $(1 + \alpha)r$. The second difference is that the call to **prune** is made with the arguments $\text{prune}(T_i, s_i, \alpha r/4, \alpha r/2 + \ell_{max})$.
3. The algorithm **prune** needs to be modified to handle the edge lengths, and the variation, called **weighted-prune** is described below.

The analysis of the weighted case mirrors the unweighted case in Section 3.3 closely. However, one big change is that we will not be concerned with the sizes of the trees, but the sum of the lengths of edges in them. We will use the same notation from Section 3.2, with the caveat that all distances are defined with respect to the edge lengths $\ell()$. Also, we will define $\ell(T)$ to be $\ell(T) = \sum_{e \in T} \ell(e)$.

LEMMA 4.1. *Let \mathcal{T}' be the collection of trees formed in any run of **explore** on a tree T rooted at vertex s_i with rope length l . Let $T' \in \mathcal{T}'$ be a tree constructed by exploring from an incomplete vertex v . If T' is incomplete, then $\Delta_{T'}(v, T') > l - d_{G^*}(s, s_i) - \Delta_T(s_i, T) - r$. Furthermore, if $l - d_{G^*}(s, s_i) - \Delta_T(s_i, T) \geq r$, then the initial tree T becomes explored.*

Proof. As in the proof of Lemma 3.1, we consider an incomplete tree T' created after running **bDFX** from some vertex v during a run of **explore**. As before, the length of the rope remaining when **explore** starts from v is $l' = l - d_{G^*}(s, s_i) - d_T(s_i, v) \geq l - d_{G^*}(s, s_i) - \Delta_T(s_i, T)$. Let $G' = (V', E')$ be the subgraph explored during the **bDFX** from v . Suppose $w \in V'$ is an incomplete vertex in T' . It must be the case that when **bDFX** started at w with a remaining rope of length d , there must have been an unexplored edge (w, w') incident on w such that $\ell(w, w') > d$. This implies that $d(v, w) + d > l'$ and therefore, $d(v, w) > l' - d \geq l' - r$.

For the second part, observe that for any incomplete vertex $v \in T$, $d(s_i, v) \leq \Delta_T(s_i, T)$, and therefore when v is visited during **explore**, the remaining rope length is at least r . Since each edge (v, v') incident on v has length at most r , all the edges incident on v will get explored after this step.

Lemmas 3.2 and 3.3 hold even for the weighted case, because the basic exploration process and the tree construction and pruning does not change. Lemma 3.4 needs to be changed slightly to reflect the weights.

LEMMA 4.2. *After pruning a tree T rooted at v , $\Delta_T(v, T)$ becomes at most $\text{maxD} - \text{minD}$, provided $\text{maxD} - \text{minD} \geq r$. If $\Delta_T(v, T) \geq \text{minD}$ initially, then after pruning also $\Delta_T(v, T) \geq \text{minD}$. For any pruned subtree T_w created with root w , $\Delta_T(w, T_w) > \text{minD}$, provided $\text{maxD} - \text{minD} \geq r$.*

Proof. Assume the first part of the lemma is not true. After the termination of function **prune**, let $u \in T$ be any vertex such that $d_T(v, u) > \text{maxD}$. Let the path P from v to u consist of vertices $v_1 = v, v_2, \dots, v_r = u$. Let j be the smallest index such that $d_T(v, v_j) \leq \text{minD}$ and $d_T(v, v_{j+1}) > \text{minD}$. If $d_T(v, v_j) = \text{minD}$, then **prune** would have pruned T at v_j , which contradicts the assumption. If $d_T(v, v_j) < \text{minD}$, $d_T(v, v_{j+1}) \leq \text{minD} + r \leq \text{maxD}$, and in this case, **prune** would have pruned T at v_{j+1} , which again contradicts the assumption.

The second and third parts of the lemma follow directly from the definition of algorithm **weighted-prune**.

The time complexity of the **weighted-CFX** algorithm is analyzed in the following theorem, whose proof closely mirrors Theorem 3.1.

THEOREM 4.1. *Any unknown weighted graph $G = (V, E)$ with source node s and (weighted) radius r can be explored by a tethered robot with rope length $(3 + \alpha)r$ for $0 < \alpha < 1$ in $\Theta(|E|/\alpha)$ edge traversals.*

Proof. As in the proof of theorem 3.1, the proof rests on showing that after the first step, each tree T with root s_i containing incomplete vertices that is explored in the **CFX** algorithm has $\Delta_T(s_i, T) \geq \alpha r/4$ and hence, length $\ell(T) \geq \alpha r/2$. This is not true for the first time **explore** is done on the initial tree consisting of just $\{s\}$, but is true for all trees created in subsequent steps.

During every successive iteration of the main loop, we choose the tree $T_i \in \mathcal{T}$ closest to s with the closest vertex $s_i \in T_i$. By Lemma 4.2, if $\Delta_{T_i}(s_i, T_i) \geq \text{minD} = \alpha r/4$, this continues to hold after the **prune** step. Also, if the subtree T_w rooted at node w is pruned, we have $\Delta_{T_w}(w, T_w) \geq \text{maxD} - \text{minD} = (1 + \alpha/4)r$. Thus, the trees that are created by the **prune** step have sufficient size and depth, and we only need to argue that the new trees T' created by **explore** have sufficient depth and size.

Consider a run of **explore**, where some tree T rooted at s_i is explored. Then, by Lemma 3.3, $d_{G^*}(s, s_i) \leq r$ and by Lemma 4.2, $\Delta_T(s_i, T) \leq \text{maxD} =$

$(1 + \alpha/2)r$. By Lemma 4.1, it follows that T gets explored fully. Let v be any node in T that was incomplete prior to calling `explore` on T , and let T' be the newly explored tree created by `bDFX` from v . If T' is incomplete, it follows from Lemma 4.1, that $\Delta_{T'}(v, T') \geq (3 + \alpha) - d_{G^*}(s, s_i) - \Delta_T(s_i, T) - r \geq \alpha r/2$.

The rest of the arguments in the proof of Theorem 3.1 hold if we use $\ell(T)$ instead of $|T|$, and that $\sum_T \ell(T) \leq 2\ell(E)$. With this, the theorem follows.

5 Simple Extension for an Unknown Radius

We now assume that the radius is not known but is discovered by the robot. Let us first look at a simple solution by starting with an initial guess r' of the radius, some constant value. We search the graph as much as possible using a rope of length $(1 + \alpha)r'$ and if this length is insufficient enough to explore the whole graph, we repeat with a rope double the length. Under this situation, since any radius $r' > r$ suffices, we obtain a running time of $O(|E| \log r)$. Observing that we can use prior information to avoid re-exploring edges yields a time of $O(|E| + |V| \log r)$.

We can, however, do much better and surprisingly we only need to modify the CFX algorithm slightly, in two places. The changes involve the two function calls in the main loop. Namely, we alter the following two lines:

```
prune( $T_i, s_i, \alpha d_{G^*}(s, s_i)/4, 9\alpha d_{G^*}(s, s_i)/16$ )
explore( $T, s_i, (1 + \alpha)d_{G^*}(s, s_i)$ )
```

As an additional caveat, notice if $d_{G^*}(s, s_i) = 0$, the robot is given a rope length of size 0 and is, therefore, unable to explore any edges. So, we actually replace $d_{G^*}(s, s_i)$ with $\max(d_{G^*}(s, s_i), c)$ for some constant $c > 0$. For the remainder of this section, we let algorithm CFX refer to this new modified version. The essentials behind the analysis for this algorithm are obviously nearly identical to the original algorithm. Again, the key is to maintain the minimum size criterion. However, in this case, we desire a more relaxed constraint that, for any tree $T \in \mathcal{T}$, $|T| \geq \alpha d_{G^*}(s, T)/4$.

Since our modification is so slight, all previous lemmas still hold for this algorithm. The only change lies in the proof of the main theorem.

THEOREM 5.1. *Any unknown graph $G = (V, E)$ with source node s and unknown radius r can be explored by a tethered robot with maximum rope length $(1 + \alpha)r$ for $0 < \alpha < 1$ in $\Theta(|E| + |V|/\alpha)$ edge traversals.*

Proof. First, we note that if $r < c$, a simple BFS search would suffice. As in the previous proof, we begin by proving that after each iteration the minimum size

criterion is maintained. In this case, we prove for every tree $T \in \mathcal{T}$, $|T| \geq \max(d_{G^*}(s, T), c)\alpha/4$. If the graph is still incomplete after the initial iteration of the loop, \mathcal{T} contains only one tree T . By Lemma 3.1, since $d_{G^*}(s, T) = 0$, $|T| \geq (1 + \alpha)c > \alpha c/4$.

Let us now examine the pruning process for the closest tree $T_i \in \mathcal{T}$ with vertex s_i . Let T be the resulting pruned tree of T_i . Since $\min D = \alpha d_{G^*}(s, s_i)/4$, Lemma 3.4 states that T still has proper size. Let us now examine more closely any pruned subtree T_w . By Lemma 3.4 $|T_w| > \max D - \min D = 5\alpha d_{G^*}(s, s_i)/16$. To see that the criterion still holds notice that

$$\begin{aligned} d_{G^*}(s, T_w) &\leq d_{G^*}(s, s_i) + d_{G^*}(s_i, w) \\ &= d_{G^*}(s, s_i)(1 + \alpha/4) \\ &< 5d_{G^*}(s, s_i)/4. \end{aligned}$$

Substituting the above equation, we see that $|T_w| > \alpha d_{G^*}(s, T_w)/4$.

Again, we now only need to show that every newly explored incomplete tree also maintains the minimum size criterion. After pruning T , Lemma 3.4 states that $\Delta_T(s_i, T) \leq \max D = 9\alpha d_{G^*}(s, s_i)/16$. Let v be any vertex in T which was incomplete prior to calling `explore` on T and let T' be the newly explored tree discovered by the `bDFX` function on v . From the `explore` procedure, we know that $l' = l - d_{G^*}(s, s_i) - d_T(s_i, v) \geq (1 + \alpha)d_{G^*}(s, s_i) - d_{G^*}(s, s_i) - \max D = 7\alpha d_{G^*}(s, s_i)/16 > 0$. Since $v \in T$ and $v \in T'$, we can see that

$$\begin{aligned} d_{G^*}(s, T') &\leq d_{G^*}(s, s_i) + d_{G^*}(s_i, v) \\ &\leq d_{G^*}(s, s_i) + 9\alpha d_{G^*}(s, s_i)/16 \\ &= 25d_{G^*}(s, s_i)/16. \end{aligned}$$

If T' is incomplete, from Lemma 3.1, we see that $|T'| > l' \geq 7\alpha d_{G^*}(s, T')/25 > \alpha d_{G^*}(s, T')/4$.

Recall that we break the robot traversal steps on a tree T into three groups, $S_1(T), S_2(T), S_3(T)$. Let $S_1(T) = 2d_{G^*}(s, s_i)$ be the number of steps taken by the first and last lines of the algorithm performing a roundtrip from s to s_i . Let $S_2(T) = 2|T|$ be the number of steps taken to perform a depth first traversal of T . Let $S_3(T)$ be the number of steps taken by all calls to `bDFX` for one call of `explore`.

By the minimum size criterion, we can bound $S_1(T) = 2d_{G^*}(s, s_i) \leq 8|T|/\alpha$. Letting \mathcal{T}_e be the set of all trees explored, we can bound the total steps \mathcal{S} taken by the updated CFX algorithm as

$$\begin{aligned} \mathcal{S} &= \sum_{T \in \mathcal{T}_e} (S_1(T) + S_2(T) + S_3(T)) \\ &\leq 2|E| + \sum_{T \in \mathcal{T}_e} (2 + 8/\alpha)|T| \\ &\leq 2|E| + (2 + 8/\alpha)2|V| \\ &= \Theta(|E| + |V|/\alpha). \end{aligned}$$

By Lemma 3.2, we know that once the algorithm terminates it has fully explored the graph. We now

only need to show that the length of the rope used is no more than $(1 + \alpha)r$. After every iteration on a tree T_i with closest vertex s_i , we use a rope of length $(1 + \alpha)d_{G^*}(s, s_i)$. By Lemma 3.3, we know that since $G^* \neq G$ there exists a vertex v such that $d_{G^*}(s, v) \leq r$. From Lemma 3.2, v belongs to some tree in \mathcal{T} . Recall that s_i is chosen to be the closest vertex among all vertices in all trees in \mathcal{T} . Therefore, $d_{G^*}(s, s_i) \leq d_{G^*}(s, v) \leq r$ completing the proof.

Using a simple reduction, we can also achieve similar bounds for the piecemeal search when the radius is unknown.

COROLLARY 5.1. *Any unknown graph $G = (V, E)$ with source node s and unknown radius r can be explored by a fueled robot using a capacity of at most $2(1 + \alpha)r$ for $0 < \alpha < 1$ in $\Theta(|E|/\alpha)$ edge traversals.*

5.1 Treasure Hunting

In the treasure hunting problem or shortest path problem, recall that we are searching for a node t at unknown distance from s in an infinite graph G^∞ . We define the graph $G^k \subset G^\infty$ to be the subgraph induced by all nodes v such that $d_{G^\infty}(s, v) \leq k$. If we let $r = d_{G^\infty}(s, t)$, in general graphs the treasure hunting problem must explore all vertices and edges in G^r . To see this, we note that the adversary only needs to make t be the last vertex in G^r visited by any exploration. If we modify algorithm CFX to stop once the vertex t is discovered, we know that the robot can never go beyond distance $(1 + \alpha)d_{G^*}(s, t) \leq (1 + \alpha)r$ from s . This observation yields the following corollary.

COROLLARY 5.2. *For any unknown infinite graph G^∞ with source node s , we can find a destination node t by exploring the subgraph $G^{(1+\alpha)d(s,t)} = (V, E)$ in $O(|E| + |V|/\alpha)$ time for $0 < \alpha < 1$.*

6 Conclusion and Open Problems

We have presented a linear algorithm for exploration of general unknown graphs and thus answered the open problem of Awerbuch, Betke, Rivest, and Singh [3]. Though computational time is not usually an issue in graph exploration, it is interesting to note that our algorithms can be implemented with $\Theta(|E|)$ computational time as well.

We conclude with some interesting open questions.

- It is critical for our algorithm to use a rope of length $(1 + \alpha)r$. It is interesting to see what is the minimum rope length which still allows linear running time. We conjecture that $r + o(r)$ is not sufficient. On the other hand, it is easy to see that

piecemeal search with a fuel capacity of $2r + o(r)$ is not possible in linear time on the “popsicle” graph consisting of a path of length $n/2$ connected to a clique on $n/2$ vertices. Let the start vertex be the first vertex on the path, so $r = n/2$. If the fuel limit is $n + f(n)$, it necessarily takes at least $\Theta(\frac{n^3}{f(n)})$ time.

- Another related question is whether a linear time algorithm exists for the treasure hunting problem of [3]. In other words, does corollary 5.2 hold with $\alpha = 0$?
- The running time of our algorithm is $2|E| + O(|V|)$. In the light of [17], it is worth exploring whether a $|E| + O(|V|)$ algorithm exists.

References

- [1] S. ALBERS AND M. R. HENZINGER, *Exploring unknown environments*, in Proceedings of the Twenty-Ninth Annual Symposium on Theory of Computing (STOC), 1997, pp. 416–425.
- [2] B. AWERBUCH, B. BERGER, L. COWEN, AND D. PELEG, *Near-linear cost sequential and distributed constructions of sparse neighborhood covers*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science (FOCS), 1993, pp. 638–647.
- [3] B. AWERBUCH, M. BETKE, AND R. M. SINGH, *Piecemeal graph learning by a mobile robot*, in Proceedings of the 8th Annual Conference on Computational Learning Theory (COLT), 1995, pp. 638–647. (Also in *Information and Computation*, 152(2):155–172, 1999).
- [4] B. AWERBUCH AND S. KOBOUROV, *Polylogarithmic-overhead piecemeal graph exploration*, in Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT), 1998, pp. 280–286.
- [5] E. BAR-ELI, P. BERMAN, A. FIAT, AND P. YAN, *Online navigation in a room*, in Symposium on Discrete Algorithms (SODA), 1992, pp. 237–249.
- [6] M. A. BENDER, A. FERNNDEZ, D. RON, A. SAHAI, AND S. P. VADHAN, *The power of a pebble: Exploring and mapping directed graphs*, vol. 176(1), 2002, pp. 1–21.
- [7] M. A. BENDER AND D. K. SLONIM, *The power of team exploration: Two robots can learn unlabeled directed graphs*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS), 1994, pp. 75–87.
- [8] P. BERMAN, A. BLUM, A. FIAT, H. KARLOFF, A. ROSÉN, AND M. SAKS, *Randomized robot navigation algorithms*, in Proceedings of the 7th Annual Symposium on Discrete Algorithms (SODA), 1996, pp. 75–84.
- [9] M. BETKE, R. RIVEST, AND M. SINGH, *Piecemeal learning of an unknown environment*, in Proceedings of the 6th Annual Conference on Computational Learning Theory (COLT), 1993. (Also in *Machine Learning*, vol.28, 2-3, February 1995).
- [10] A. BLUM AND P. CHALASANI, *An online algorithm for improving performance in navigation*, in Proceedings of

- the Thirty-Fourth Annual Symposium on Foundations of Computer Science (FOCS), 1994, pp. 2–11.
- [11] A. BLUM, P. RAGHAVAN, AND B. SCHIEBER, *Navigating in unfamiliar geometric terrain*, in Proceedings of 23rd ACM Symposium on Theory of Computing (STOC), 1991, pp. 494–504.
- [12] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, MIT Press/McGraw-Hill, 1990.
- [13] X. DENG, T. KAMEDA, AND C. H. PAPADIMITRIOU, *How to learn an unknown environment*, in Proceedings of the 32nd Symposium on Foundations of Computer Science (FOCS), 1991, pp. 298–303.
- [14] X. DENG AND C. H. PAPADIMITRIOU, *Exploring an unknown graph*, in Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS), 1990, pp. 355–361. (Also in *Journal of Graph Theory*, 32(3):265–297, 1999).
- [15] P. FRAIGNAUD, L. GASIENIEC, D. KOWALSKI, AND A. PELC, *Collective tree exploration*, in LATIN, 2004.
- [16] F. HOFFMANN, C. ICKING, R. KLEIN, AND K. KRIEGEL, *A competitive strategy for learning a polygon*, in Proceedings of the 8th Annual Symposium on Discrete Algorithms (SODA), 1997, pp. 166–174.
- [17] P. PANAITIE AND A. PELC, *Exploring unknown undirected graphs*, in Proceedings of the 9th Annual Symposium on Discrete Algorithms (SODA), 1998, pp. 316–322. (Also in *Journal of Algorithms*, 33:281–295, 1999).
- [18] C. PAPADIMITRIOU AND M. YANAKAKIS, *Shortest paths without a map*, Theoretical Computer Science, 84 (1991), pp. 127–150.
- [19] N. S. V. RAO, S. KARETI, W. SHI, AND S. S. IYENGAR, *Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms*, tech. rep., July 1993. ORNL/TM-12410.
- [20] R. L. RIVEST AND R. E. SCHAPIRE, *Inference of finite automata using homing sequences*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC), 1989, pp. 411–420. (Also in *Information and Computation*, 103(2):299–347, 1993).

```

CFX( $s, r, \alpha$ )
  Initially the only tree is  $\{s\}$ 
  Repeat
    Let  $T_i$  be the closest subtree to  $s$  in  $G^*$ 
    Let  $s_i$  be the closest vertex in  $T_i$  to  $s$ 
    prune( $T_i, s_i, \alpha r/4, \alpha r/2$ )
      producing  $\mathcal{T}_i = \{T_{i_0}, T_{i_1}, \dots, T_{i_k}\}$ 
    // Here  $T_i = \cup_{T \in \mathcal{T}_i} T$ 
     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T_i\} \cup \mathcal{T}_i$ 
    Let  $T \in \mathcal{T}_i$  such that  $s_i \in T$ 
    explore( $T, s_i, (1 + \alpha)r$ )
    remove all explored trees from  $\mathcal{T}$ 
    merge any trees in  $\mathcal{T}$  with common vertices
  Until  $\mathcal{T} = \emptyset$ 

prune( $T, v, \text{minD}, \text{maxD}$ )
  If  $T = \{s\}$ , return
  Root  $T$  at  $v$ 
   $\mathcal{T}_i \leftarrow \{T\}$ 
  for each vertex  $w \in T$  such that  $d_T(v, w) = \text{minD}$ 
    Let  $T_w \subseteq T$  be the subtree of  $T$  rooted at  $w$ 
    if  $\Delta_T(v, T_w) > \text{maxD}$ 
      // Separate subtree  $T_w$  from  $T$ 
       $T \leftarrow T \setminus T_w$ 
       $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{T_w\}$ 

explore( $T, s_i, l$ )
  MOVE ROBOT from  $s$  to  $s_i$  via shortest known path
  MOVE ROBOT using depth first traversal on  $T$ 
  for each incomplete vertex  $v$  visited
    Let  $l'$  be the length of rope remaining
    Call bDFX( $v, l'$ )
    Let  $E'$  be the set of new explored edges
    Let  $V'$  be the set of vertices in  $E'$ 
    Let  $\mathcal{T}'$  be a spanning forest of  $G' = (V', E')$ 
     $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}'$ 
  MOVE ROBOT BACK from  $s_i$  to  $s$  retracing old path

```

Figure 2: The CFX algorithm with start node s and rope length $(1 + \alpha)r$

```

weighted-bDFX( $v, l$ )
  if  $v$  is tagged
    return
  if all edges incident at  $v$  are explored or
   $l < \ell(e)$  for each unexplored edge  $e$  incident on  $v$ 
    // Maximum length reached. Must backtrack
    if  $v$  has any unexplored edge
      Mark  $v$  as incomplete
    else
      Mark  $v$  as explored
    return
  Mark  $v$  as tagged
  For each unexplored edge  $(v, w) \in E$  with  $d(v, w) \leq l$ 
    MOVE ROBOT to  $w$  along edge  $(v, w)$ 
    Mark  $(v, w)$  as explored
    Call bDFX( $w, l - d(v, w)$ )
    MOVE ROBOT BACK from  $w$  to  $v$  along edge  $(v, w)$ 
  Mark  $v$  as explored

```

Figure 3: The recursive bDFX algorithm from node v with a remaining rope length of l

```

prune( $T, v, \text{minD}, \text{maxD}$ )
  Root  $T$  at  $v$ 
   $\mathcal{T}_i \leftarrow \{T\}$ 
   $\forall$  node  $w \in T$ , let  $T_w$  be the subtree rooted at  $w$ 
  while  $\exists w$  s.t.  $d(v, w) \leq \text{maxD} - \text{minD}$ 
  and  $\Delta_{T_w}(w, T_w) \geq \text{minD}$ 
    // Separate subtree  $T_w$  from  $T$ 
     $T \leftarrow T \setminus T_w$ 
     $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{T_w\}$ 

```

Figure 4: The weighted-prune algorithm.