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
Max-Planck-Institut für Informatik
Saarbrücken, Germany
kumar@mpi-sb.mpg.de

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, 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. 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. Thus, a trivial lower bound for the running time of an exploration algorithm measured by the number of edge traversals is $|E| + |V|$. In the remainder of this paper we assume that the graph to be explored is connected, which implies an exploration lower bound of $|E|$. For the unconstrained exploration problem the classical search algorithms, breadth-first search (BFS) and depth-first search (DFS), e.g. see [11], are optimal within a constant factor.

The piecemeal search model, introduced by Betke, Rivest and Singh [8], adds two natural constraints to the graph exploration problem:

- *Continuity:* the robot must traverse a continuous path in the graph. 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 even more restricted problem of graph exploration: exploration by a tethered robot with a rope of length l. An algorithm for the more constrained tethered robot exploration can easily be modified to an algorithm for the piecemeal exploration. In the tethered robot model, the robot is tied to the start node by a tether (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

*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.

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. Section 2.1 shows that the piecemeal search problem with a fuel tank of size $2(1+\beta)r$ can be reduced with only a constant factor increase in time to a tethered robot problem with a rope of length $(1+\alpha)r$ where $\beta > \alpha$.

Note that neither the piecemeal 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* [17]). The exploration can be modeled geometrically as shown by Blum, Raghavan, and Schieber [10], Bar-Eli, Berman, Fiat, and Yan [5], Deng and Papadimitriou [12] and Hoffmann *et al* [14]. Papadimitriou and Yanakakis [16] consider finding shortest paths in various unknown graphs and Blum and Chalasani [9] give a "k-trip" shortest path algorithm.

The exploration can also be modeled with a finite state automata as shown by Rivest and Schapire [18] or as either a directed or undirected graph. The exploration is modeled by a directed graph by Deng and Papadimitriou [13] and Albers and Henzinger [1]. Bender and Slonim [6] consider the problem with two cooperating robots. Randomized robot navigation was addressed by Berman *et al* [7].

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 [15] address the problem of minimizing the constants in such linear time exploration, achieving a bound of $|E| + \Theta(|V|)$.

The problem of piecemeal exploration of an undirected graph is described by Betke, Rivest, and Singh [8] 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 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. The exploration using a robot with limited fuel tank capacity is a simple reduction from the tethered robot case and runs in $\Theta(|E|/\alpha)$ steps. The exploration time is optimal within a constant factor. If the radius is not known, the solution can also be extended to work in $\Theta(|E| + |V|/\alpha)$ and $\Theta(|E|/\alpha)$ time for tethered and piecemeal explorations respectively. This problem is similar to 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.

In Section 2 we present the `CFX` algorithm which optimally solves the piecemeal graph exploration problem and analyze its running time. In Section 3 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 4 concludes with a few open problems.

2 Efficient Graph Exploration

Let s be the *start node* of the graph exploration and 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. In both variations of the exploration problem (piecemeal search and tethered robot search), the radius must be known in advance. Otherwise the fuel tank or the rope length may be insufficient to reach the farthest vertex. In fact, for a fueled robot to explore a graph it must have tank size at least $2 * r$. Similarly, a tethered robot must have rope length at least r. As the fueled robot is a less constrained version, we concentrate our efforts on the tethered robot exploration and assume that the robot has a rope length of size $(1+\alpha)r$ for some $\alpha > 0$.

2.1 Piecemeal Search Reduction

Let us first describe the reduction from piecemeal search to a tethered search. We can transform an algorithm

```
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) ∈ 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

A that performs a tethered robot exploration, with a rope of length $(1+\alpha)r$, in time T to an algorithm B to perform a piecemeal exploration with a tank of size $C \geq 2(1+\beta)r, \beta > \alpha$ in time $\frac{1+\beta}{\beta-\alpha}T$ in the following manner [3]. B emulates A for every $2(\beta-\alpha)r$ steps and then goes for a refuelling and returns. Since algorithm A ensures that the robot is always within distance $(1+\alpha)r$ from the start node, the maximum fuel C' used before the robot reaches the start node for refueling is $C' \leq 2(\beta-\alpha)r + 2(1+\alpha)r = C$. Since every refueling operation taking time $2(1+\alpha)r$ is accompanied by $2(\beta-\alpha)r$ emulated steps from algorithm A, algorithm B takes time $T + T(1+\alpha)/(\beta-\alpha) = T(1+\beta)/(\beta-\alpha)$.

2.2 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 [15]. However, if the length of the rope is $O(r)$, there are simple cases in which the

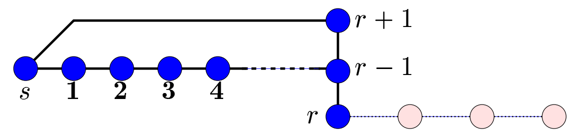

Figure 2: An example of a bad graph traversal of bDFX. The nodes are labelled by their depth first traversal order. Since node r has depth r at the time visited, the grey nodes cannot be explored. After exploring edge $(s, r+1)$, we discover a path between s and r with length only 3. However, in bDFX, these edges $(r+1, r-1)$ and $(r-1, r)$ cannot be traversed again. Thus, the grey nodes must remain unexplored.

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. For convenience, we also call any graph $G' = (V', E')$ *explored* when all of its vertices are explored and *incomplete* if there are any incomplete vertices. Notice that the unexplored edges making a vertex in V' incomplete may not belong to E' but to the original edge set E.

The recursive bDFX algorithm simply starts at a node v with a given rope length l and recursively visits all unvisted 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 2. After termination, the algorithm may leave many edges *unexplored*.

2.3 Optimal Tethered Exploration

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 $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of vertex disjoint subtrees of G whose union consists of 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 $\mathcal{T}$. The tree is then split into several smaller subtrees $T_{i_0}, T_{i_1}, \ldots T_{i_k}$, and the resulting subtrees replace T_i in $\mathcal{T}$. The idea is to "prune" the trees maintaining 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 $T_{k+1}, T_{k+2}, \ldots$ which are also added to $\mathcal{T}$.

At every stage, we also maintain the complete known subgraph $G^* \subseteq G$ consisting of all edges and vertices explored or incomplete so far. For any graph $H = (V, E)$, let $P_H(v, w)$ be the shortest path between v and w in H. Let $d_H(v, w) = |P_H(v, w)|$ represent the distance between vertices v and w in H, i.e., the number of *edges* in the path $P_H(v, w)$. We also expand our definition to include the distance between sets of vertices $U, W \subseteq V$. Let $P_H(U, W)$ be the shortest path in H between any $u \in U$ and $w \in W$. Similarly, we define $d_H(U, W) = \min_{u \in U, w \in W} d_H(u, w)$ to be the *minimum* distance between the two sets in H. In the same way, let $\Delta_H(U, W) = \max_{u \in U, w \in W} d_H(u, w)$ be the *maximum* distance. 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.

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. For any tree T rooted at a vertex v, we have that $|T| > \Delta_T(v, T)$. After each phase, we wish to maintain one very important *minimum-size criterion* stating that for all $T \in \mathcal{T}$, $|T| \geq \alpha r/4$.[1]

Figure 3 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, 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 $\Delta_{T_w}(w, T_w) \geq \alpha r/4$.

```
CFX(s, r, α)
  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, αr/4, αr/2)
      producing 𝒯_i = {T_i0, T_i1, ... T_ik}
    // Here T_i = ∪_{T∈𝒯_i} T
    𝒯 ← 𝒯 \ {T_i} ∪ 𝒯_i
    Let T ∈ 𝒯_i such that s_i ∈ T
    explore(T, s_i, (1+α)r)
    remove all explored trees from 𝒯
    merge any trees in 𝒯 with common vertices
  Until 𝒯 = ∅

prune(T, v, minD, maxD)
  Root T at v
  𝒯_i ← {T}
  for each vertex w ∈ T such that d_T(v, w) = minD
    Let T_w ⊆ T be the subtree of T rooted at w
    if Δ_T(v, T_w) > maxD
      // Separate subtree T_w from T
      T ← T \ T_w
      𝒯_i ← 𝒯_i ∪ {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'
      Compute a spanning tree T' of G' = (V', E')
      𝒯 ← 𝒯 ∪ {T'}
  MOVE ROBOT BACK from s_i to s retracing old path
```

Figure 3: The CFX algorithm with start node s and rope length $(1 + \alpha)r$

2.4 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.

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

LEMMA 2.1. *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 $l' = l - d_{G^*}(s, s_1) - d_T(s_i, v) > 0$. 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 2.2. *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 tree 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 2.3. *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 2.4. *After pruning a tree T rooted at v, the maximum depth of T is less than or equal to* `maxD`*. In other words, $\Delta_T(v, T) \leq$* `maxD`*. If $|T| \geq$* `minD`*, then after pruning, $|T| \geq$* `minD`*. For any pruned subtree T_w created, $|T_w| >$* `maxD` *−* `minD`*.*

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 $d_T(v, u) >$ `maxD`. Let w be the ancestor of u such that $d_T(v, w) =$ `minD`. We now have $\Delta_T(v, T_w) >$ `maxD`. 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 $|T'| <$ `minD`. Let $w \in T$ be the root of any pruned subtree, T_w. Since w was chosen to be pruned, it necessarily follows that $d_T(v, w) =$ `minD`. Let u be the parent of w in T. Let $P = P_T(v, u)$. Since u cannot be pruned, $P \subseteq T'$. This, however, implies that $|T'| \geq |P| + 1 =$ `minD`.

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

THEOREM 2.1. *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. For simplicity, we make the distinction that $\alpha < 1$. Clearly, if $\alpha > 1$, we are given a larger rope length and can explore more. We first show that the minimum size criterion is maintained after each 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 2.1, $|T| \geq (1 + \alpha)r > \alpha r/4$.

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 `minD` $= \alpha r/4$, Lemma 2.4 shows that T still maintains the minimum size criterion.

Furthermore, Lemma 2.4 also shows that for each pruned subtree T_w, $|T_w| > \mathtt{maxD} - \mathtt{minD} = \alpha r/4$. 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 2.3, we see that $d_{G^*}(s, s_i) \leq r$. Lemma 2.4 states that $\Delta_T(s_i, T) \leq \mathtt{maxD} = \alpha r/2$. 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 $l' = l - d_{G^*}(s, s_i) - d_T(s_i, v) \geq (1+\alpha)r - r - \mathtt{maxD} = \alpha r/2 > 0$. If T' is incomplete, from Lemma 2.1, 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 2.1, 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. 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 2.3 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. The tree T is removed but these vertices may remain as explored vertices on other trees in $\mathcal{T}$. 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

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

By Lemma 2.2, 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.

3 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:

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

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 3.1. *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 2.1, 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 $\texttt{minD} = \alpha d_{G^*}(s,s_i)/4$, Lemma 2.4 states that T still has proper size. Let us now examine more closely any pruned subtree T_w. By Lemma 2.4 $|T_w| > \texttt{maxD} - \texttt{minD} = 5\alpha d_{G^*}(s,s_i)/16$. 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 2.4 states that $\Delta_T(s_i,T) \leq \texttt{maxD} = 9\alpha d_{G^*}(s,s_i)/16$. 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 $l' = l - d_{G^*}(s,s_i) - d_T(s_i,v) \geq (1+\alpha)d_{G^*}(s,s_i) - d_{G^*}(s,s_i) - \texttt{maxD} = 7\alpha d_{G^*}(s,s_i)/16 > 0$. 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 2.1, 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

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

By Lemma 2.2, 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 2.3, we know that since $G^* \neq G$ there exists a vertex v such that $d_{G^*}(s,v) \leq r$. From Lemma 2.2, 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 3.1. *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.*

3.1 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 3.2. *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 $0<\alpha<1$.*

4 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 3.2 hold with $\alpha = 0$?

- The running time of our algorithm is $2|E| + O(|V|)$. In the light of [15], 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)*, pages 416–425, 1997.

[2] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Near-linear cost sequential and distribured constructions of sparse neighborhood covers. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 638–647, 1993.

[3] B. Awerbuch, M. Betke, and R. R. M. Singh. Piecemeal graph learning by a mobile robot. In *Proceedings of the 8th Annual Conference on Computational Learning Theory (COLT)*, pages 638–647, 1995. (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)*, pages 280–286, 1998.

[5] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. On-line navigation in a room. In *Symposium on Discrete Algorithms (SODA)*, pages 237–249, 1992.

[6] 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)*, pages 75–87, 1994.

[7] 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)*, pages 75–84, 1996.

[8] M. Betke, R. Rivest, and M. Singh. Piecemeal learning of an unknown environment. In *Proceesings of the 6th Annual Conference on Computational Learning Theory (COLT)*, 1993. (Also in Machine Learning, vol.28, 2-3, February 1995).

[9] A. Blum and P. Chalasani. An online algorithm for improving performance in navigation. In *Proceesings of the Thirty-Fourth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2–11, 1994.

[10] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proceedings of 23rd ACM Symposium on Theory of Computing (STOC)*, pages 494–504, 1991.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[12] 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)*, pages 298–303, 1991.

[13] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. In *Proceedings of the 31st Symposium on Foundations of Computer Science (FOCS)*, pages 355–361, 1990. (Also in *Journal of Graph Theory,* 32(3):265–297, 1999).

[14] 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)*, pages 166–174, 1997.

[15] P. Panaite and A. Pelc. Exploring unknown undirected graphs. In *Proceedings of the 9th Annual Symposium on Discrete Algorithms (SODA)*, pages 316–322, 1998. (Also in *Journal of Algorithms,* 33:281–295,1999).

[16] C. Papadimitriou and M. Yanakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[17] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical report, Oak Ridge National Laoratory, July 1993. ORNL/TM-12410.

[18] 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)*, pages 411–420, 1989. (Also in *Information and Computation,* 103(2):299–347, 1993).