A Polynomial Time Algorithm for Exploring Unknown Graphs with Deficiency d

Stephen Kwek*
Department. of Computer Science
Washington University
St. Louis, MO 63130
kwek@cs.wustl.edu

Abstract

We present an $O(dn^2+m)$-time algorithm for exploring (constructing) an unknown strongly connected graph G with m edges and n vertices by traversing at most dn^2+m edges. Here, d is the minimum number of edges needed to add to G to make it Eulerian. This parameter d was introduced by Deng and Papadimitriou in (Deng & Papadimitriou 1990) and is known as the deficiency of a graph. They showed that in the worst case, $\Omega\left(d^2m\right)$ edge traversals are required and gave an algorithm that achieves an upper bound of $d^{O(d)}m$ edge traversals. Subsequently, Albers and Henzinger (Albers & Henzinger 1997) gave an algorithm that achieves an upper bound of $O\left(d^6d^{2\log d}m\right)$ edge traversals. Our bound is an improvement over these earlier bounds when $d=\omega(\log n)$.

Introduction

The Graph Exploration Problem. We look at the following classic *graph exploration* problem: A robot has to *explore* (*i.e.*, construct) an unknown terrain represented by a strongly connected directed multi-graph G, by repeatingly select an outgoing edge from its current vertex (position) and traverse it. The robot knows all the traversed (*seen*) edges and visited nodes, and can recognize them when they are encountered again. In other words, at each stage of the exploration, the robot knows a proper subgraph G', consisting of the traversed edges, of G. However, the robot does not know:

- how many nodes or edges are there in the graph G,
- where each unseen edge leads to, and
- the origin of any unseen edge of a visited node.

The robot is said to have *learned* the graph G if all the edges have been traversed (and thus, the graph G' is the same as G). Since the graph is strongly connected, it is trivial that the robot can learn any graph and knows that its exploration task is complete when all of the edges in G' have been traversed. The goal is to learn the graph by traversing as few edges as possible.

It is easy to show that if the graph is Eulerian, then the robot only needs to traverse at most $4m$ edges (Deng & Papadimitriou 1990) where m is the number of edges. In fact, Deng and Papadimitriou (Deng & Papadimitriou 1990) showed that the number of edge traversals is related to the number of edges, d, that are needed to make a graph Eulerian $G(V,E)$. They call this quantity the *deficiency* of G. Deng and Papadimitriou showed that in the worst case, the number of edge traversals required can be $\Omega(d^2m)$ and gave an algorithm that traverses $O(d^{O(d)}m)$ edges. Recently, Albers and Henzinger (Albers & Henzinger 1997) gave a better algorithm that achieves an upper bound of $O\left(d^6d^{2\log d}m\right)$ edge traversals. Our bound is better than these earlier bounds when $d=\omega(\log n)$.

In this paper, we present a simple polynomial-time algorithm, which employs a depth-first search strategy, that traverses at most $O(dn^2+m)$ edges where n is the number of vertices. Note that if the unknown graph is dense say $m=\Omega(n^2)$ edges then our bound becomes $O(dm)$. However, it can be shown easily that any reasonable algorithm traverses at most mn edges. Thus for sparse graph, say $m=O(n)$, the worst case bounds on the number of edge traversals for all three algorithms are mn. Notice that mn could be $\Omega(2^dm)$ if $d=O(\log n)$.

This problem has also been investigated by Koenig and Smirnov (Keonig & Smirnov 1996) where they considered directed graphs obtained from undirected graphs by replacing each edge with a pair of edges, one for each direction. Notice that such digraphs are always Eulerian and hence can be explored by traversing each edge at most four times (Deng & Papadimitriou 1990).

Other Related Work. Motivated by robotics, various related graph exploration (navigation) problems have been investigated by both theoreticians and AI practitioners. Frequently, geometric constraints of the

*The author is supported by NSF NYI Grant CCR-9357707 (of Sally Goldman) with matching funds provided by Xerox PARC and WUTA.

environment are ignored and the environment is assumed to be represented by a graph. The research conducted by AI practitioners is mostly empirical in their approach and we refer the reader to (Korf 1990; Pemberton & Korf 1992; Stentz 1995; Smirnov *et al.* 1996).

Theoretical studies of graph exploration problems were first initiated by Papadimitriou and Yannakakis (Papadimitriou & Yannakakis 1991) where they were interested in the problem of finding a shortest path between a pair of points under various graph theoretic and geometric settings. Motivated by this earlier work, Blum *et. al.* considered the exploration problem in the Euclidean plane with convex obstacles (Blum, Raghavan, & Schieber 1991) and other variations (Blum, Raghavan, & Schieber 1991; Blum & Chalasani 1993).

The exploration problem examined in this paper is due to Deng and Papadimitriou (Deng & Papadimitriou 1990). Subsequently, they together with Kameda (Deng, Kameda, & Papadimitriou 1991) studied a geometric version of this problem where the robot's task is to explore a room with polygonal obstacles and the goal is to minimize the total distance traveled. Recently, Berman *et. al.* (Berman *et al.* 1996) investigated the case where the obstacles are oriented rectangles. Other work relating to exploring geometric domain are be found in (Lumelsky & Stepanov 1986; 1987; E. Bar-Eli & Yan 1992; Lumelsky & Tiwari 1994; Taylor & Kriegman 1994; Angluin, Westbrook, & Zhu 1996).

Betke *et. al.* (Betke, Rivest, & Singh 1995) and Awerbuch *et. al.* (Awerbuch *et al.* 1995) investigated the problem with the additional constraints that the robot is required to return to its starting point for 'refueling' periodically. Bender and Slonim (Bender & Slonim 1994) illustrated how two robots can collaborate in exploring directed graph with regular degree and indistinguishable vertices.

Preliminary. Let $in(v)$ and $out(v)$ denote the indegree and outdegree of the vertex v. We say a vertex is *deficient* if its indegree is greater than its outdegree. The *deficiency* of a vertex v, $d(v)$, is

$$d(v) = \begin{cases} in(v) - out(v) & \text{if } v \text{ is deficient} \\ 0 & \text{if } v \text{ is not deficient} \end{cases}$$

The minimum number of edges needed to add to a graph G to make it Eulerian is called the *deficiency* of G and is denoted here by d. Clearly,

$$d = \sum_{v \in V} d(v).$$

We say that a vertex is *exhausted* if the robot has traversed all its outgoing edges. If the robot reaches an exhausted vertex v then we say the robot is *stuck* at v.

```
EXPLORE(s)
1   stack := ∅
2   V(G') = {s}
3   E(G') = ∅
4   u := s
5   do
6       do
7           if u is not exhausted then
8               traverse an unexplored edge (u, v)
9               V(G') := V(G') ∪ {v}
10              E(G') = E(G') ∪ {(u, v)}
11              push(v)
12              u := v
13      until stuck at u (i.e., u is exhausted)
14      do
15          y := pop(stack)
16      until stack = ∅ or y is not exhausted
17      if (stack ≠ ∅) then
18          find a shortest path from u to y in G'
19          traverse the path so that the robot is at y
20          u := y
21  until stack = ∅
22  return G'
```

Figure 1: An algorithm for exploring an unknown graph that traverses at most $O(dn^2 + m)$ edges.

A Simple Depth First Search Strategy

Suppose a is the vertex where the robot begins its exploration. To simplify our discussion, we assume the graph the robot is exploring is a 'modified' graph obtained by adding a new vertex s with an edge (s, a) and the robot begins at s. Note that although the modified graph is no longer strongly connected, it is still possible for the robot to traverse all the edges. With this assumption, the vertices along any path that begins with s have even degree except the last vertex and s.

Our strategy EXPLORE is shown Figure 1. It simply traverses an unexplored edge when there is one (Step 6 to 13) until the robot is stuck. As the robot traverses the edges in this greedily manner, we push the current vertex into a stack (Step 11).

When the robot is stuck, we simply pop the stack until either the stack is empty or the poped vertex y is not exhausted. In the latter case, we claim (see Claim 2) that there is a path from u that leads to y in the graph G' obtained by the robot's exploration so far. The robot then traverses this path to y and repeat the greedy traversal of the unexplored edges (by letting u be y and repeat Steps 6 to 13). In the former case, we show below that the robot has traversed all the edges of G.

Since we pop a vertex out of the stack only if it is exhausted, we have the following observation about the vertices in G' that have not been exhausted.

Observation 1 *The vertices in G' that are not ex-*

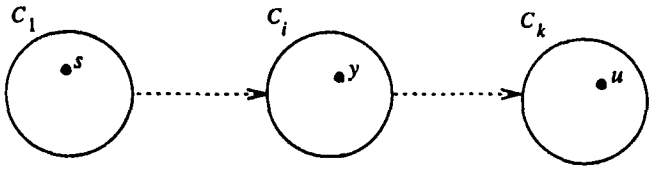

Figure 2: The graph G_P induced by P

hausted are in the stack.

Thus, if we can show that (see Claim 2) there is a path connecting u to the unexhausted vertex y in G' at Step 18, then we are sure that the robot is able to traverse an unseen edge whenever there is an unexhausted vertex in G'. Thus, by Observation 1, when the stack is empty, all the vertices in G' are exhausted. Since G is strongly connected, the latter implies that the robot has completed it's exploration of G.

Claim 2 *In Step 18, there is a path from u to y in G'.*

Proof: The stack obtained just after the execution of the loop in Steps 6 to 13, induces a path P from s to u which passes through y (maybe several times). Observe that the vertices from the last occurrence (on P) of y to u in the path P are all exhausted except y. Consider the graph G_P induced by P. It is clear that (see Figure 2) G_P is a graph with strongly connected components C_1, ..., C_k such that there is exactly one edge going from C_i to C_{i+1} and u is in C_k. If y is also in C_k as u then there is a path from u to y.

Thus, suppose y is in C_i such that $i \neq k$. Consider the graph G'. If there is no path in G' from u to any vertex in $C_j, j \leq i$, then the vertices in G' reachable by u cannot reach the vertices in $C_1, \cdots, C_i$. By Observation 1, only the vertices in $C_1, \cdots, C_i$ can be unexhausted, thus, there is no path from u to y in the unknown graph G. This contradicts the strongly connectedness of G. □

In the following, we derived an upper bound on the number of edge traversals by EXPLORE. The proof involves assignment of labels to the edges as we traverse them. This label assignment is to facilitate the analysis of the algorithm and has nothing to do with the correctness of the algorithm.

Clearly, the first vertex u where the robot is stuck in Step 13 has to be a deficient vertex. The sequence of vertices which we poped from the stack in Steps 14 to 16 induces a path P' in G from u to y. Give the edges in P' excluding those lying on a cycle a label '1' and set a counter D to 2. Subsequently, each execution of Step 14 to 16 induces a path P' from u to y. We label the edges in P' according to the following rules:

Rule 1: Suppose the robot is stuck at a vertex u where there is a label i such that there are more[1] outgoing edges with label i than incoming edges with label i. Then we add to each edge in P' excluding those lying on a cycle (in P') a label i. If there are more than one such i, EXPLORE arbitrarily selects one such i.

Rule 2: Suppose the condition in Rule 1 does not hold. It is clear that for the vertex u, the number of incoming traversed edges exceeds the number of traversed outgoing edges (and all outgoing edges have been traversed). That is, u must be a deficient vertex. In this case, we label the edges in P' excluding those lying on a cycle (in P') the value of D and increment D by 1.

Notice that a new label is created only when Rule 2 is applied. This implies that u is deficient and we can 'blame' the activation of Rule 2 on 1 unit of G's deficiency. Thus, we have the following observation.

Observation 3 *There are at most d labels and Rule 2 is invoked at most d times.*

Lemma 4 *For each vertex u and each label i, Rule 1 is applied at most once. In other words, each edge is assigned at most d (distinct) labels.*

Proof: Consider the following two cases when an outgoing edge of u is labeled i for the first time:

Case 1: u is exhausted. In this case, We assign an incoming edge of u a label of i and continue popping the stack. Subsequently, whenever a new outgoing edge of u is labeled i, an unlabeled incoming edge of u is labeled i. Thus, for the vertex u, the number of incoming edges that are labeled i is the same as the number of outgoing edges. That is, in the future, Rule 1 will never be applied to u with label i.

Case 2: u is not exhausted. In this case, we apply Rule 2. We stop popping the stack and unlike Case 1, we do not assign a label of i to any of u's incoming edges. The robot then continue to explore an unseen outgoing edge of u. Notice that for all the other vertices, the number of incoming edges with label i is the same as the number of outgoing edges with label i. Subsequently, no new edge is labeled i until Rule 1 is applied to u with label i. When this happens, there is exactly one incoming edge and one outgoing edge of u that are labeled i. Note that u is now exhausted and we can apply similar argument as in Case 1.

□

Observation 3 and Lemma 4, imply that the robot is stuck at most dn times. Each time the robot is stuck, it traverses at most n traversed edges to reach an unexhaust vertex in Step 18. The robot traverses seen edges only in Step 18 when it is stuck.

We can view the popping of a vertex from the stack in step 14 as poping an edge and each edge can be shown to poped exactly once. Moreover, by Observation 3, Step 18 is executed at most dn time and each

[1] In fact, it is easy to see that in this case, there is exactly one more outgoing edge than incoming edges with label i.

execution of Step 18 takes $O(n)$. Thus we have the following theorem.

Theorem 5 *The number of edges traversals by* EXPLORE *is at most* $dn^2 + m$ *and the time complexity is* $O(dn^2 + m)$.

Future Directions

The algorithm EXPLORE presented in this paper is based on a simple depth first search strategy. The analysis is quite straightforward and lax. However, it indicates a possibility that a variant or tighter analysis of EXPLORE may give rise to a better bound, possibly metting the lower bound of Deng and Papadimitriou (Deng & Papadimitriou 1990) on the number of edge traversals which warrants further investigation. Although it is unlikely that EXPLORE outperforms known algorithms (see (Korf 1990; Pemberton & Korf 1992; Stentz 1995; Smirnov *et al.* 1996)) that are used in practice, we hope that our analysis will provide new insight to this classical problem which gives us a more efficient strategy.

Notice that EXPLORE does not assume that the robot knows how many unseen edges are leading into or out of the visited vertex. Such information may be available in practice and may result in more efficient exploration algorithm.

Another research direction is to consider using multiple robots to explore the graph. It is interesting to know if the total number of edge traversals can be made smaller than using a single robot.

Acknowledgements

The author thanks Sally Goldman, Monika Henzinger, Subhash Suri and Stephen Scott for helpful conversations regarding this work.

References

Albers, S., and Henzinger, M. 1997. A subexponential algorithm for exploring an unknown graph. In *Proc. 29th Annu. ACM Sympos. Theory Comput.* To appear.

Angluin, D.; Westbrook, J.; and Zhu, W. 1996. Robot navigation with range queries. In *Proc. 28th Annu. ACM Sympos. Theory Comput.* 469-478.

Awerbuch, B.; Betke, M.; Rivest, R.; and Singh, M. 1995. Piecemeal graph exploration by a mobile robot. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, 321–328. ACM Press, New York, NY.

Bender, M. A., and Slonim, D. K. 1994. The power of team exploration: two robots can learn unlabeled directed graphs. In *Proceedings of the 35rd Annual Symposium on Foundations of Computer Science*, 75–85. IEEE Computer Society Press, Los Alamitos, CA.

Berman, P.; Blum, A.; Fiat, A.; Karloff, H.; Rosen, A.; and Saks, M. 1996. Randomized robot navigation algorithms. In *Proceedings SODA 96.* to appear.

Betke, M.; Rivest, R. L.; and Singh, M. 1995. Piecemeal learning of an unknown environment. *Machine Learning* 18(2/3):231–254.

Blum, A., and Chalasani, P. 1993. An on-line algorithm for improving performance in navigation. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, 2–11. IEEE Computer Society Press, Los Alamitos, CA.

Blum, A.; Raghavan, P.; and Schieber, B. 1991. Navigating in unfamiliar geometric terrain. In *Proc. 23th Annu. ACM Sympos. Theory Comput.*, 494–504. ACM.

Deng, X., and Papadimitriou, C. H. 1990. Exploring an unknown graph. In *Proc. 31th Annu. IEEE Sympos. Found. Comput. Sci.*, volume I, 355–361.

Deng, X.; Kameda, T.; and Papadimitriou, C. 1991. How to learn in an unknown environment. In *Proc. of the 32nd Symposium on the Foundations of Comp. Sci.*, 298–303. IEEE Computer Society Press, Los Alamitos, CA.

E. Bar-Eli, P. Berman, A. F., and Yan, P. 1992. Online navigation in a room. In *Proc. 3rd SODA*, 75–84.

Keonig, S., and Smirnov, Y. 1996. Graph learning with a nearest neighbor approach. In *Proceedings of the 9th Conference on Computaitonal Learning Theory*, 19–28.

Korf, R. 1990. Real-time heuristic search. *Artificial Intelligence* 42(3):189–211.

Lumelsky, V., and Stepanov, A. 1986. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Trans. on Automatic Control* AC-31:1059–1063.

Lumelsky, V., and Stepanov, A. 1987. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shapes. *Algorithmica* 2:403–430.

Lumelsky, V., and Tiwari, S. 1994. An algorithm for maze searching with azimuth input. In *IEEE Conference on Robotics and Automation*, 111–116.

Papadimitriou, C., and Yannakakis, C. 1991. Shortest path wothout a map. *Theoretical Computer Science* 84:127–150.

Pemberton, J., and Korf, R. 1992. Incremental path planning on graphs with cycles. In *Proceedings of the AI Planning Systems Conference*, 179–188.

Smirnov, Y.; Keonig, S.; Veloso, M.; and Simmons, R. 1996. Efficeint goal-directed exploration. In *Proceedings of the National Conference on AI.* 292-297.

Stentz. 1995. The focussed d^* algorithm for real-time replanning. In *Proceedings of the International Joint Conf. on AI*, 1652–1659.

Taylor, C. J., and Kriegman, D. J. 1994. Vision-based motion planning and exploration algorithms for mobile robots. In *Proc. of the Workshop on Algorithmic Foundation of Robotics.*