Random Walks and Undirected Graph Connectivity: A Survey

Anna R. Karlin *　　Prabhakar Raghavan†

March 31, 1994

Abstract

We survey a number of algorithms that decide connectivity in undirected graphs. Our focus is on the use of random walks as a tool in reducing the space complexity of these algorithms.

The *undirected s–t* connectivity (USTCON) problem is the following: given an undirected graph G and two vertices s and t in G, decide whether s and t are in the same connected component. Throughout this paper, n will denote the number of vertices in the graph, and m will denote the number of edges. The USTCON problem is important in the study of space-bounded complexity classes.

In examining algorithms for solving this problem, there are two resource measures of interest.

1. *Time* is measured by the number of steps taken by the algorithm on a standard model of computation such as the unit-cost RAM [1].

*DEC Systems Research Center, 130 Lytton Ave., Palo Alto, CA 94301. karlin@src.dec.com

†IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. pragh@watson.ibm.com

2. *Space* is the number of bits of storage used by the algorithm. Here we do not include the read-only storage in which the input to the problem is given.

A classical algorithm for USTCON is *breadth-first search.* Initially, every node of the graph except s is unmarked. Next, we mark s and place it into a queue. Thereafter, the algorithm proceeds by removing the node at the head of the queue and exploring its neighbors one at a time. Each neighbor that is unmarked is now marked, and added to the queue. The algorithm terminates with output YES when t is marked, or with output NO when the queue is empty and t has not been marked. Since each edge is examined at most once in each direction, the time used by the algorithm is clearly $O(m)$. Space is used for marking nodes, and is thus $O(n)$.

Aleliunas, Karp, Lipton, Lovász and Rackoff [3] proposed an alternative algorithm for USTCON using random walks. At each step, the algorithm proceeds from a vertex of G to a neighbor of the vertex chosen uniformly at random. The walk starts at s, and lasts $4m(n-1)$ steps. If t is visited in the course of the walk, the algorithm outputs YES; otherwise, it outputs NO. Note that when this algorithm outputs NO, it may do so erroneously. We proceed to bound the probability of such an error.

Let $\mathcal{C}_u(G)$ denote the expected length of a walk that starts at u and ends upon visiting every vertex in G at least once. Let $\mathcal{C}(G)$ be the *cover time* of G, defined by $\mathcal{C}(G) = \max_u \mathcal{C}_u(G)$. Aleliunas *et al.* [3] proved the following theorem (see also Göbel and Jagers [11]):

Theorem 1

$$\mathcal{C}(G) \leq 2m(n-1).$$

Proof: Let T be any spanning tree of G. There is a traversal of T, going through vertices $v_0, v_1, \ldots, v_{2n-2} = v_0$ that traverses each edge of T exactly once in each direction. Further, every vertex of G appears at least once in the sequence $v_0, v_1, \ldots, v_{2n-2}$. Consider a random walk that starts at v_0 and terminates upon returning to v_0, having visited the vertices $v_1, v_2, \ldots$ in that sequence. Since this walk has visited every vertex in G, an upper bound on the expected length of this walk is an upper bound on $\mathcal{C}_{v_0}(G)$. Let H_{xy} be the number of steps until state y is first visited starting at state x Then

$$\mathcal{C}_{v_0}(G) \leq \sum_{j=0}^{2n-3} H_{v_j,v_{j+1}} = \sum_{\text{edges } (u,w)\in T} H_{uw} + H_{wu}.$$

It is known [2, 3, 8, 11] that

$$H_{vw} + H_{wv} \leq 2m, \quad (vw) \in T.$$

But this upper bound holds no matter which vertex of T (and therefore G) we designated to be v_0 in the traversal we chose; therefore $\mathcal{C}(G) \leq 2m(n-1)$. $\square$

By Theorem 1 and the Markov inequality, we have:

Corollary 2 *The probability that the random walk algorithm errs when outputting NO is at most* $1/2$.

The random walk algorithm uses space to remember its position in the graph, and to count up to $4m(n-1)$; clearly $O(\log n)$ bits suffice for both of these tasks.

Before proceeding to study the space-time products of the algorithms we have discussed, we pause to observe a feature of the proof of Theorem 1 that will be of interest later on. Chandra *et al.* [8] showed that for any pair of vertices v, w in G,

$$H_{vw} + H_{wv} = 2mR_{vw}, \tag{1}$$

where R_{vw} is the *effective resistance* between v and w in an electrical network obtained by replacing each edge of G by a unit resistor. Define the distance between v and w to be the effective resistance between them. Let $R_{span}(G)$ denote the weight of the minimum spanning tree in G according to this distance metric. Following the above proof of Theorem 1 and (1), it is clear that $\mathcal{C}(G) \leq 2mR_{span}(G)$. We now return to the space-time product of our algorithms.

Interestingly, both the breadth-first search algorithm and the random walk algorithm have the same space-time product, $O(mn)$, within a logarithmic factor. Aleliunas *et al.* [3] posed the following question: "The reachability problem for undirected graphs can be solved in logspace and $O(mn)$

time by a probabilistic algorithm that simulates a random walk, or in linear time and space by a conventional deterministic graph traversal algorithm. Is there a spectrum of time-space trade-offs between these extremes?" The remainder of this paper will be devoted to this question.

For simplicity of presentation, we introduce the following notation. We say that $f(n) = \tilde{O}(g(n))$ if there exist positive numbers c and N such that, for all $n \geq N$, $|f(n)| \leq \log^c n |g(n)|$. Thus, the space-time product achieved by both the above algorithms is $\tilde{O}(mn)$.

Broder, Karlin, Raghavan and Upfal [6] took the first step towards answering the question posed by Aleliunas *et al.* Their algorithm is based on the following observation: the fraction of "wasted" steps of the random walk (steps in which the walk revisits previously visited vertices) increases as the walk grows longer. Hence many short random walks are more efficient in exploring a graph than one long random walk.

The scheme proposed by Broder *et al.* is the basis for the algorithms we discuss below. We now outline their scheme, using space $\tilde{O}(p)$. The algorithm consists of repeating the following steps $O(\log n)$ times.

1. Choose p vertices of G at "random" and place a marker on each. Call these vertices, together with s and t, *leaders.*

2. Repeat $O(\log n)$ times: Take a random walk of length τ from each leader. If such a walk connects two leaders, then mark them and all the other leaders known to be connected to them, as belonging to the same component. If at any point s and t are marked as being in the same component, then stop and output YES.

If in the course of the algorithm, s and t are not marked as being in the same component, output NO. Clearly the space used by the algorithm is $\tilde{O}(p)$. The time requirement is $\tilde{O}(p\tau)$.

The random choice in Step 1 and the value of τ are what distinguish the three algorithms we discuss:

- Algorithm 1: The p vertices are chosen according to the stationary distribution of the random walk, and τ is set to be $\tilde{O}(m^2/p^2)$.

- Algorithm 2: $p/2$ vertices are chosen according to the stationary distribution, and $p/2$ are chosen uniformly at random. τ is set to be $\tilde{O}(m^{3/2}n^{1/2}/p)$.
- Algorithm 3: $p/2$ vertices are chosen according to the stationary distribution, and $p/2$ are chosen in inverse proportion to vertex degrees. τ is set to be $\tilde{O}((m \sum_v \text{degree}(v))/p)$.

There are two key facts that must be proven in order to show that these algorithms work. The first is to show that a set of p random walks of length τ, one from each of the randomly chosen leaders, visits all the vertices of a connected graph with high probability. Otherwise an adversary could choose s and t among those vertices unlikely to be visited from the other leaders and conceivably foil the algorithm. In other words, we need to derive a bound on the expected time required by p parallel and independent random walks to cover the graph, a problem of interest in its own right. This involves proving something about short-term behavior of the Markov chain and coverage of local neighborhoods in a graph.

The second fact to prove is that if s and t are in the same component and enough leaders are chosen within that component, then with high probability s and t are linked up after a small number of walks from each leader. Coverage of the graph as described above does not assure linkage, since s and t may be visited only by walks from two disjoint sets of leaders that are never linked. Furthermore all the vertices in G could be visited by the walks even with s and t in different components.

Broder *et al.* (algorithm 1) choose a vertex of G to be a leader in proportion to its degree. It follows that each of the random walks of length τ begins in the stationary distribution. They also set $\tau = \tilde{O}(m^2/p^2)$. From this they obtain a space-time product of $\tilde{O}(m^2)$.

To aid the intuition of the reader, let us consider the case when G is a simple path on n vertices. For p leaders chosen at random, with high probability, the maximum gap between two leaders is no more than $n \ln n/p$; the expected time to cover this maximum gap is $\Theta(n^2 \log^2 n/p^2)$. Hence $O(\log n)$ trials (random walks of length $O(n^2 \log^2 n/p^2)$ from each leader) will almost surely cover all the gaps between them for a total of $\Theta(n^2 \log^3 n/p)$ steps. Thus each leader "discovers" its closest neighbor leader in both directions, and therefore all leaders are marked as being in the same component.

For the analysis below, we need to look at the random walk in two ways: first, as a Markov chain $X(t)$ where each state is a vertex in G (the vertex process); second, as a Markov chain $Y(t)$ where each state is a directed edge (the edge process). The transition rule for the vertex process is that if $X(t) = v$, then $X(t+1)$ is equally likely to be any of the neighbors of vertex v. The edge process is defined by $Y(t) = [X(t-1), X(t)]$, $t \geq 1$. The stationary distribution of the vertex process, denoted π, is given by $\pi_v = d_v/(2m)$ where d_v is the degree of the vertex v, and the stationary distribution of the edge process, denoted π', is given by $\pi'_{[v,w]} = 1/(2m)$.

In order to prove the above key facts, we show that the following stronger properties hold with high probability.

1. The random walks from the leaders (each of length τ) will together visit every edge of the graph.
2. A random walk of length τ from any edge will hit some leader.

Property 1 clearly implies the first key fact above. From Properties 1 and 2 together, it is possible to prove the second key fact. Intuitively, the algorithm is likely to discover that two leaders a and b are in the same connected component because there is an edge e such that a random walk of length τ from a is likely to hit e, and a random walk of length τ from e is likely to hit b.

The probability that an edge e is visited in a single random walk of length τ is bounded from below as follows.

$$E_\pi[\text{Number of visits to } e \text{ in } \tau \text{ steps}] = \tau/m$$

$$\leq Pr[e \text{ visited}] \cdot E_e[N_e(\tau)],$$

where $E_e[N_e(t)]$ denotes the expected number of times that a random walk of length t starting at e returns to e. Thus,

$$Pr[e \text{ visited}] \geq \frac{\tau/m}{E_e[N_e(\tau)]},$$

and the problem boils down to upper bounding $E_e[N_e(\tau)]$. Using standard results in renewal theory [2] and an elegant result of Carne [7], Broder *et al.* proved that

$$E_e[N_e(\tau)] \leq \tau/m + \tilde{O}(\sqrt{\tau}).$$

Setting τ to a value that is $\tilde{O}(m^2/p^2)$ yields a space-time product that is $\tilde{O}(m^2)$ for all values of p of interest. The random choices of the leaders and the value of τ chosen by Broder *et al.* cannot hope to achieve a space-time product that is $\tilde{O}(mn)$, as the following example shows. Consider the "barbell" graph consisting of two cliques of size $n/3$ each, joined by a path of length $n/3$. Suppose that $p = n^{3/4}$, and that s and t are in different cliques. In order to achieve the space-time product $\tilde{O}(mn)$, τ must be $\tilde{O}(mn/p^2) = \tilde{O}(n^{3/2})$. A random walk of length $\tilde{O}(n^{3/2})$ from one clique is very unlikely to reach the other clique. On the other hand, no leaders are likely to be selected from the vertices on the path.

The above example highlights the need to favor vertices of low degree when selecting leaders. Barnes and Feige [4] (algorithm 2) took the first step in this direction by selecting $p/2$ leaders according to the stationary distribution (as in [6]), and the remaining $p/2$ uniformly at random. They were able to achieve a space-time product that is $\tilde{O}(m^{3/2}n^{1/2})$.

Feige [10] (algorithm 3) achieved the space-time product that is $\tilde{O}(m\hat{R})$, where $\hat{R} = \sum_v \text{degree}(v)$. Since $\hat{R} \leq n/d_{min}$, where d_{min} is the minimum vertex degree in G, this space-time product is $\tilde{O}(mn/d_{min})$. In his algorithm $p/2$ leaders are chosen according to the stationary distribution, and $p/2$ are chosen in inverse proportion to vertex degrees. The key step in Feige's analysis is to show that

$$E_e[N_e(\tau)] = \tilde{O}(\sqrt{\tau \hat{R}/m}). \tag{2}$$

There are three crucial points to proving (2). The first is an observation made in [4], that Feige's leader-distribution scheme on G can be analyzed by considering the Broder *et al.* scheme on a modified graph H. The difference between H and G is that self loops are introduced in H.

The second point is to use a method developed by Aldous [2], in his proof that on regular graphs, $E_v[N_v(t)] = O(\sqrt{t})$. Aldous' technique can be applied to the analysis of $E_e[N_e(t)]$ on arbitrary graphs, and can take into account parameters such as d_{min}.

The third point is that $\hat{R}$ is a property of a graph as a whole, whereas we are analyzing properties of short random walks that only visit a small fraction of the graph. Feige's pebble distribution scheme and careful analysis allows him to show that $\hat{R}$ is in fact reflected in the properties of short random walks.

Is there scope for improving Feige's space-time product of $\tilde{O}(m\hat{R})$? Of related interest is the question: how well does the quantity $m\hat{R}$ approximate $\mathcal{C}(G)$? Coppersmith, Feige and Shearer [9] have shown that $R_{span}(G)$ and $\hat{R}$ are within constant factors of each other, for every connected graph. Thus, Feige achieves a space-time product that matches that achieved by the Aleliunas *et al. upper bound* for the random walk. However, there are graphs for which this upper bound is weak. On these, Feige's choice of τ results in a space-time product that is weaker than that of the random walk algorithm.

On the other hand, Feige's algorithm can do better than breadth-first search. For example, on graphs that are regular of degree cn, breadth-first search achieves a space-time product that is $\Omega(n^3)$. In contrast, Feige's algorithm achieves a space-time product that is $\tilde{O}(n^2)$ on such a graph.

What are the intrinsic limits to algorithms that choose leaders at random and then take short random walks from each leader? Feige [10] shows that a class of leader distribution schemes including all those described here cannot achieve a space-time product that is $o(mD)$, where D is the sum of the diameters of the connected components of G. For a survey of other negative results, the reader is referred to Borodin [5]. Wigderson [12] gives a comprehensive survey of additional related results.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] D.J. Aldous. Reversible Markov chains and random walks on graphs, 1993. Unpublished Monograph, Berkeley.

[3] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze

problems. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 218–223, San Juan, Puerto Rico, October 1979.

[4] G. Barnes and U. Feige. Short random walks. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 728–737, 1993.

[5] A. Borodin. Time-space tradeoffs. In *Proceedings of the 4th Annual ISAAC*, pages 209–220. Springer-Verlag, 1993.

[6] A.Z. Broder, A.R. Karlin, P. Raghavan, and E. Upfal. Trading space for time in undirected s-t connectivity. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 543–549, Seattle, WA, May 1989. ACM.

[7] T. K Carne. A transmutation formula for Markov chains. *Bull. Sci. Math.*, 109:399–405, 1985.

[8] A. K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 574–586, Seattle, May 1989.

[9] D. Coppersmith, U. Feige, and J. Shearer. Random walks on regular and irregular graphs. Technical Report CS93-15, The Weizmann Institute of Science, 1993.

[10] U. Feige. A randomized time-space tradeoff of $\tilde{O}(m\hat{R})$ for USTCON. In *34th Annual IEEE Symposium on Foundations of Computer Science*, pages 238–247, 1994.

[11] F. Göbel and A.A. Jagers. Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336, 1974.

[12] A. Wigderson. The complexity of graph connectivity. Unpublished manuscript, Computer Science Dept., Hebrew University, 1993.