Time-Space Tradeoffs for Undirected Graph Traversal *

Paul Beame † Allan Borodin ‡ Prabhakar Raghavan §
Walter L. Ruzzo † Martin Tompa † §

Abstract

We prove time-space tradeoffs for traversing undirected graphs. One of these is a quadratic lower bound on a deterministic model that closely matches the recent probabilistic upper bound of Broder, Karlin, Raghavan, and Upfal. The models used are variants of Cook and Rackoff's "Jumping Automata for Graphs".

1 The Complexity of Graph Traversal

Graph traversal is a fundamental problem in computing, since it is the natural abstraction of many search processes. In computational complexity theory, graph traversal (or more precisely, *st*-connectivity) is a fundamental problem for an additional reason: understanding the complexity of directed versus undirected graph traversal seems to be the key to understanding the relationships among deterministic, probabilistic, and nondeterministic space-bounded algorithms. For instance, although directed graphs can be traversed nondeterministically in polynomial time and logarithmic space simultaneously, there is evidence that they cannot be traversed deterministically in polynomial time and small space simultaneously (Tompa [19]). In contrast, *undirected* graphs can be traversed in polynomial time and logarithmic space *probabilistically* by using a random walk (Aleliunas *et al.* [1], Borodin *et al.* [9]); this implies similar resource bounds on (nonuniform) deterministic algorithms (Aleliunas *et al.* [1]).

A more detailed inspection of the time and space requirements of undirected graph traversal algorithms follows. Depth-first or breadth-first search can traverse any n vertex, m edge undirected graph in $O(m+n)$ time, but requires $\Omega(n)$ space. Alternatively, a random walk can traverse an undirected graph using only $O(\log n)$ space, but requires $\Theta(mn)$ expected time [1]. In fact, Broder *et al.* [12] have shown recently that there is a spectrum of compromises between time and space for this problem: any graph can be traversed in space S and expected time T, where $ST = O(m^2 \log^5 n)$. This raises the intriguing prospect of proving that logarithmic space and linear time are not simultaneously achievable or, more generally, proving a time-space tradeoff that closely matches these upper bounds.

Although it would be desirable to show a tradeoff for a general model of computation such as a Turing machine or random access machine, obtaining such a tradeoff is beyond the reach of current techniques. Thus it is natural to consider a "structured" model (Borodin [8]), that is, one whose basic move is based on the adjacencies of the graph, as opposed to one whose basic move is based on the bits in the graph's encoding. An appropriate structured model for proving such a tradeoff is some variant of the JAG ("jumping automaton for graphs") of Cook and Rackoff [13]. Such an automaton has a finite control, and a limited supply of pebbles that it can move from vertex to adjacent vertex ("walk") or directly to a vertex containing another pebble ("jump"). The purpose of its pebbles is to mark certain vertices temporarily, so that they are recognizable when some other peb-

*This material is based upon work supported in part by the Natural Sciences and Engineering Research Council of Canada, by the National Science Foundation under Grants CCR-8703196, CCR-8858799, and CCR-8907960, and by IBM under Research Contract 16980043.

†Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, WA, U.S.A. 98195

‡Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

§IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY, U.S.A. 10598

ble reaches them. Blum and Sakoda [6] and Blum and Kozen [5] considered similar models.

The pebbles represent vertex names that a structured algorithm might record in its workspace. Walking represents replacing that vertex name by some adjacent vertex found in the input. Jumping represents copying a previously recorded vertex name.

The JAG is a structured model, but not a weak one. In particular, it is general enough to encompass most known algorithms for graph traversal. For instance, a JAG can execute a depth-first or breadth-first search, provided it has one pebble for each vertex, by leaving a pebble on each visited vertex in order to avoid revisiting it, and keeping the stack or queue of pebble names in its finite control. Furthermore, as Cook and Rackoff point out, a JAG with the additional power to move a pebble from vertex i to vertex $i+1$ can simulate an arbitrary Turing machine.

The time T used by a JAG is the number of pebble moves. Cook and Rackoff define the space to be $S = P\log_2 n + \log_2 Q$, where P is the number of pebbles and Q the number of states of the automaton. (Keeping track of the location of each pebble requires $\log_2 n$ bits of memory, and keeping track of the state requires $\log_2 Q$.) Using this model, they prove that traversal of *directed* graphs requires space $\Omega(\log^2 n/\log\log n)$, closely matching the upper bound of Savitch [18]. Their lower bound has been extended to randomized JAGs by Berman and Simon [4].

In this paper we use variants of the JAG to study the tradeoff between time and space for the problem of *undirected* graph traversal.

Several authors have previously considered undirected graph traversal by a JAG with an unlimited number of states but only one pebble, a model better known as a *universal traversal sequence* (Aleliunas *et al.* [1], Alon *et al.* [2], Bar-Noy *et al.* [3], Bridgland [11], Istrail [16], Karloff *et al.* [17]). Borodin, Ruzzo, and Tompa [10] proved that such an automaton requires $\Omega(m^2)$ time (on regular graphs with $3n/2 \le m \le n^2/6 - n$). Thus, for the particularly weak version of logarithmic space corresponding to the case $P = 1$, a quadratic lower bound on time is known.

The known algorithms and the lower bounds for universal traversal sequences suggest that the true time-space product for undirected graph traversal is approximately quadratic. The main results of this paper are three lower bounds for variants of the JAG that provide progress toward proving this conjecture and, in fact, establish it for one variant. These results are outlined below.

The upper bound of $ST = O(m^2\log^5 n)$ by Broder *et al.* [12] is established on a model that is actually a restricted variant of the JAG. In their algorithm, the JAG initially drops $P-1$ pebbles on random vertices, after which they are never moved. It then uses its last pebble to explore the graph, with the others as fixed landmarks. In Section 3, using essentially the same variant of the JAG, we prove a lower bound of $PT = \Omega(n^2)$ for 3-regular graphs, independent of the value of Q. This nearly matches the upper bound, since $m = O(n)$ for 3-regular graphs. The main difference between the models is that the last pebble moves probabilistically in the upper bound model, but deterministically in the lower bound model.

The result above is the desired quadratic lower bound, on a model that is natural but more restricted than we would like. In particular, it would be nice to extend the result to a model in which all pebbles are movable. In fact, our proof does extend to give a nonlinear lower bound when some motion of the pebbles is allowed, but the bound degenerates when the pebbles are allowed to move with complete freedom. Such models are surprisingly powerful. Nevertheless, in Section 4 we prove a lower bound on a model with freely moving pebbles, but without the ability to jump one pebble to another. (This nonjumping model is closer to the one studied by Blum and Sakoda [6] and Blum and Kozen [5].) More specifically, using a very different and more complex argument, we prove that there is a family of $3P$-regular graphs whose traversal with P pebbles requires time $\Omega((Pn\log n)/\log P)$, independent of the value of Q. In particular, for constant P the time must be $\Omega(n\log n)$. Although this is not the desired quadratic lower bound, it does establish that logarithmic space and linear time are not simultaneously achievable on the nonjumping model when $m = \omega(n)$.

The results described above have the strength that they hold independent of the magnitude of Q, the number of states. Presumably the bounds can be strengthened by also accounting for Q. It is tempting to tackle first the case in which Q is constant; indeed, Cook and Rackoff [13] studied JAGs on undirected graphs in this case, showing for example that $PQ = O(1)$ is impossible. For a nonjumping variant of JAGs, in Section 5 we prove the stronger result that $PQ = \Omega(n)$ for 2-regular graphs, no matter how much time the automaton is allowed. Thus, for logarithmic space, lower bounds on time are only interesting when the number of states grows at least linearly with the size of the graph. As one simple consequence, this makes the lower bounds harder to prove, as one cannot simply make the graph so large compared to Q that the automaton is guaranteed to loop forever among some states. As a byproduct, we show that a universal traversal sequence for 2-regular graphs cannot consist solely of the repetition of a short sequence.

2 Graph-Traversing Automata

Consider the set of all n-vertex, edge-labeled, undirected graphs $G = (V, E)$ with maximum degree d. For this definition, edges are labeled as follows. For every edge $\{u, v\} \in E$ there are two labels $l_{u,v}, l_{v,u} \in \{0, 1, \ldots, d-1\}$ with the property that, for every pair of distinct edges $\{u, v\}$ and $\{u, w\}$, $l_{u,v} \neq l_{u,w}$. The problem we will be considering is "undirected st-connectivity": given G and two distinguished vertices s and t, determine if there is a path from s to t.

A *JAG* [13] is a finite automaton with Q states and P distinguishable pebbles, and whose program may depend on n and d. Two vertices s and t of its input graph are distinguished. The P pebbles are initially placed on s. In each move, based on the current state, which pebbles coincide on vertices, which pebbles are on s and t, and the edge labels emanating from the pebbled vertices, the automaton changes state, and selects some pebble p and either some $i \in \{0, 1, \ldots, d-1\}$ or some $j \in \{1, 2, \ldots, P\}$. In the former case, p is moved along the edge with label i; in the latter case, p "jumps" to the vertex occupied by pebble j. (A small difference between our version of the JAG and Cook and Rackoff's is that they allowed multiple pebbles to move in the same step. This is because they were interested in measuring space alone, whereas we are interested in measuring time as well.) A JAG that determines st-connectivity is required to enter an accepting state if and only if there is a path from s to t.

There are a number of interesting variants of JAGs. For instance, in Sections 4 and 5 we will disallow jumping. We will distinguish this nonjumping variant by referring to it as a WAG ("walking automaton for graphs"). We will also distinguish among three types of pebbles, "active", "passive", and "unmovable". The automaton as described in the previous paragraph has active pebbles, in the sense that any pebble can move; this is the model used in Section 4. A weaker notion is that of the passive pebble, which cannot move unless accompanied by an active pebble. In this case, we allow one active pebble accompanied by any number of passive pebbles to walk or jump each move. This is the model used in Section 5.

Closely related to the passive pebble is the unmovable pebble, which is placed on the graph before the automaton begins its traversal, and cannot be moved at all. This is the model discussed in Section 3. What remains is to describe the mechanism by which these unmovable pebbles are placed on the graph. The intent is to model precomputation of the input such as is done by Broder *et al.* [12] in their initial random pebble placement. Of course, such precomputation must be restricted so as to preclude solving st-connectivity itself. Therefore, the unmovable pebbles are placed based on complete knowledge of the local, but not global, structure of the graph.

Specifically, let $N_\rho(G)$ denote a list $G_1, G_2, \ldots G_n$ of graphs, each with a distinguished vertex, such that G_i is isomorphic to the radius ρ neighborhood of vertex i in G, and the isomorphism maps G_i's distinguished vertex to vertex i. For instance, $N_1(G)$ is equivalent to an ordered list of the degrees of G's vertices. Then an automaton *with P' unmovable pebbles placed by ρ-precomputation* is a pair (f, M), where M is one of the JAG variants as described above, and f is an arbitrary function

mapping $N_\rho(G)$ to $U \subseteq \{1, 2, \ldots, n\}$ with $|U| = P'$. Given G, the P' unmovable pebbles are placed on $f(N_\rho(G))$, and then M is run on the resulting pebbled graph.

3 A Lower Bound for Unmovable Pebbles

The undirected st-connectivity algorithm of Broder *et al.* [12] in outline operates as follows. First, s and t are marked by pebbles. Then $P-3$ other pebbles are placed on the graph at random. In particular, each pebble is placed on a vertex chosen at random with probability proportional to the degree of the vertex, independent of all other pebbles. These $P-1$ pebbles are not subsequently moved. The one remaining pebble then executes a small number of short random walks from each of the $P-1$ fixed pebbles. At the end of each walk, the movable pebble jumps to one of the fixed pebbles. Connectivity information is inferred from the pebbles encountered during these short walks. Broder *et al.* show that, in time $O(m^2 \log^5 n/P)$, if s and t are in the same connected component, the algorithm will discover this with high probability. Note that this algorithm can be executed on a model that is a JAG with 1-precomputation, except that its unmovable pebbles are placed randomly, and its one active pebble is allowed to move randomly.

In this section, we prove an $n^2/(2^{O(\rho)}P)$ lower bound on the time for st-connectivity on a JAG with ρ-precomputation. Our model is weaker than that used by Broder *et al.* in that our active pebble moves deterministically rather than probabilistically. On the other hand, ours is stronger in that pebble placement is determined arbitrarily (rather than probabilistically) based on complete knowledge of the local structure of the graph.

Theorem 1 *Let M be any JAG with 1 active pebble and $P-1$ unmovable pebbles placed by ρ-precomputation. Suppose M determines st-connectivity for all 3-regular n-vertex graphs. Then M requires time at least $n^2/(2^{O(\rho)}P)$.*

Proof: In this abstract, we only prove the lower bound for $\rho = 1$, corresponding to the upper bound of Broder *et al.* The generalization to arbitrary ρ is sketched at the end.

The proof generalizes the main lower bound technique introduced by Borodin *et al.* [10]. Assume without loss of generality that n is a multiple of 4. We define a family of n vertex graphs, each formed by joining two copies of an $n/2$ vertex graph H by "switching" some combination of edge pairs. We will show that M must frequently walk from one pebble to another via some distant switchable edge.

Many graphs H would work for our purposes; for definiteness, we use the $n/2$ vertex "squirrel cage": two $n/4$ vertex cycles, with each vertex on one cycle joined by a "rung" to the corresponding vertex on the other cycle. Fix any numbering of the vertices and labeling of the edges of H. Take as the set of "switchable" edges any $r = n/4-1$ of the rungs. As in Borodin *et al.* [10], for each $x \in \{0,1\}^r$ the graph G_x is formed from two copies H^0 and H^1 of H by "switching" the edges corresponding to the 1's in x. That is, if $\{u^0, v^0\}$ is the i^{th} switchable edge in H^0 and $\{u^1, v^1\}$ is the corresponding edge in H^1, then G_x has the pair of edges $\{u^b, v^{b \oplus x_i}\}, b \in \{0,1\}$. Choose s to be any vertex in H^0 and t any vertex in H^1. Let $\mathcal{G} = \{G_x \mid x \in \{0,1\}^r\}$. Notice that the only graph in $\mathcal{G}$ with no path from s to t is G_{0^r}.

The key property of the family $\mathcal{G}$ of graphs is that a walk on G_x that encounters no unmovable pebble is identical to such a walk on H, except that the walk switches from one copy of H to the other when a switched edge is crossed. After any sequence of edge crossings starting from a vertex v, M's active pebble will be in the same copy of H as v just in case the net parity with respect to x of all edge crossings is even, where the parity with respect to x of an individual edge e is defined to be x_i if e is the i^{th} switchable edge, for any $1 \le i \le r$, and 0 for all unswitchable edges.

The choice $f(N_1(G_x))$ of unmovable pebble locations is made without knowledge of which edges have been switched, since all members of $\mathcal{G}$ are 3-regular. (Note that the choice of pebble locations could even be allowed to depend on knowledge of the family $\mathcal{G}$.) Assume that the distinguished vertices s and t are marked by two extra unmovable pebbles. Note that M gains information about connectivity only by *walking* between

two pebbles; nothing is learned (directly) by jumping to a pebble. M's behavior on *all* members of $\mathcal{G}$ can be summarized by considering its behavior on G_{0^r}. Imagine that both copies of each pebbled vertex are "marked" in G_{0^r}. Now run M on G_{0^r}. Suppose in the process that there are w sequences of edges traversed while walking from one marked vertex to the next one encountered. If there exists any $x \neq 0^r$ such that the net parity with respect to x of edges crossed in each of the w sequences is even, then M encounters exactly the same pebbles in traversing G_x as it would in traversing G_{0^r}, and in particular fails to walk between pebbles in different copies of H. Thus, for M to distinguish between G_{0^r} and every other G_x, it must be that the corresponding homogeneous system of w linear equations in r unknowns over $GF(2)$ has no nonzero solution.

Now suppose some k switchable edges occurred in fewer than k of these equations. Set the (variables corresponding to the) other $r - k$ switchable edges to 0, and these k to some nonzero solution, which must exist in a homogeneous system with fewer equations than unknowns (Herstein [15, Corollary to Theorem 4.3.3]). Since such a nonzero solution cannot occur, every set of k switchable edges must occur in at least k equations. Hence, by Hall's Theorem [14], there must be a system of distinct representatives for the switchable edges. That is, for each switchable edge we can select a unique sequence containing it. Thus the total length of all the sequences must be great enough that for each switchable edge there is a unique traversal from some marked vertex. The average distance from a switchable edge to the nearest marked vertex is $\Omega(n/P)$. Hence, M's running time must be $\Omega(rn/P) = \Omega(n^2/P)$.

This construction can be generalized to a family $\mathcal{G}$ of graphs in which switched edges do not alter the local structure within any fixed radius ρ. This is done by choosing a 3-regular graph R whose girth is at least $2\rho + 1$ and whose size is $2^{O(\rho)}$ (Bollobás [7, Chapter 3]), and then constructing the half-size graph H by connecting $n/2^{O(\rho)}$ copies of R in a cycle. □

Any graph in the family $\mathcal{G}$ built from squirrel cage graphs as above can be traversed in linear time by an automaton with 2 pebbles, one of them passive, even without jumping, provided the passive pebble can be moved freely. Thus, stronger proof techniques are necessary for freely moving pebbles. This is the subject of the next section.

4 A Lower Bound for Active Pebbles

In this section we prove a time–space tradeoff for automata with P active pebbles, but no jumping. The proof generalizes an unpublished construction of Szemerédi [communicated to us by Sipser], which proved an $\Omega(n \log n)$ lower bound on the length of universal traversal sequences.

Theorem 2 *Let M be any WAG with $P \leq n^{1/41}$ active pebbles. Suppose M determines st-connectivity for all $3P$-regular n-vertex graphs. Then M requires time $\Omega((Pn \log n)/\log P)$.*

Proof: The idea underlying the proof is to build a graph with many copies of some fixed gadgets, each with many "entry points." Since M doesn't have enough pebbles to mark all the gadgets it has explored, it must spend a lot of time re-exploring each gadget from different entry points, or it risks the possibility that one of them might never be fully explored.

Imagine "growing" the graph as follows. At a general point in the construction, the graph consists of some gadgets (to be described below) that are fully specified except for the interconnections among their "entry point" vertices. Simulate M on this partial graph until it attempts to move some pebble p out of an entry point using a label for which no edge is yet defined. Our main freedom in the construction is the choice of the other endpoint of this interconnecting edge e. We want to pick it so that M will spend time $\tau = \Omega((\log n)/\log P)$ "exploring" the gadget reached through e. If we can achieve this for most of the interconnecting edges, of which there will be $\Theta(Pn)$, we will obtain an $\Omega(Pn\tau)$ time lower bound, as desired.

The interconnecting edge is chosen as follows. For simplicity, first suppose the next τ steps by M are all moves of the same pebble p, assuming p doesn't encounter another pebble. Then p will

follow some fixed path σ of length τ. In this case it would suffice to choose e to connect to any pebble-free gadget having σ as a subgraph from its entry point. It is important that the chosen gadget be pebble-free, since if p encountered another pebble, it might well deviate from the planned path σ. In particular, it might try to cross another as yet undefined connecting edge long before τ steps are spent.

Of course, the next τ moves by M might not be so simple. For example, p might make a few moves, then pause while some other pebble p' follows the same edges, encounters p, a third pebble follows part of the same path but then diverges from it, etc. Furthermore, pebbles elsewhere in the graph might also try to cross undefined connecting edges during this interval. Thus, in the general case, our lookahead scheme is much more complex than the example sketched in the previous paragraph. To accommodate diverging pebbles, the path σ in general must be replaced by a tree. Furthermore, we use an amortized cost analysis to ensure that each of the many partially defined trees simultaneously being built contributes cost τ before its connecting edge is committed.

The gadgets used need the following properties. For each labeled P-leaf tree σ of size (actually, external path length) at most τ, there is a gadget of size $O(P\tau)$ such that, for each of the gadget's $\Theta(P\tau)$ entry points v, σ is a subgraph of the gadget with σ's root at v. Each entry point accommodates $2P$ connecting edges.

We will now present the construction in more detail. We actually define a sequence of graphs G_i, $0 \leq i \leq P\lfloor n/49 \rfloor$, representing successive phases of the construction. Each graph is connected, and consists of:

- A set of gadgets, each with $L = \Theta(P\tau)$ entry vertices and a fully defined internal structure and labeling. Each vertex that is not an entry vertex has degree $3P$. Each entry vertex has P edges to neighbors in the gadget, and $2P$ *connecting edges* (see below).

- A set of labeled *committed* connecting edges joining entry vertices of gadgets. G_i will have exactly i committed connecting edges.

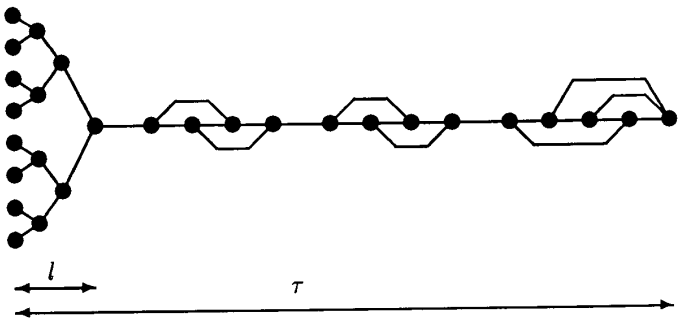

Figure 1: The "Rake" Graph

- A set of partially labeled *uncommitted* connecting edges, each joining an entry vertex u of some gadget to the root v of a complete $3P$-ary partially labeled tree Σ_v of height τ. The set of labels of the $3P$ parent-to-child edges at a given vertex of Σ_v is the set $\{0, \ldots, 3P-1\}$; all tree edges are unlabeled in the other direction, as is the uncommitted edge in the v-to-u direction.

Recall that our goal is to define a fully labeled, $3P$-regular graph G. The trees in the intermediate graphs G_i are only partially labeled, of higher maximum degree, and not degree-regular. They serve, however, as a useful artifice in the construction, since M's behavior in a tree is especially simple. In particular, note that M always moves along edge labels in $\{0, \ldots, 3P-1\}$, so all pebble motions in Σ_v will be in the parent-to-child direction. We will gradually eliminate the trees by "committing" previously uncommitted connecting edges to appropriately chosen gadgets instead.

Next we define the gadgets we will use. Figure 1 illustrates the basic idea. The graph therein, called a "rake", has a "fanin" part, a complete binary tree of height l, and a "handle" part, a chain of $\tau - l$ vertices. We will choose τ and l so that $\tau - l \geq 5$ and $\tau - l \equiv 1 \pmod 4$, and add extra edges as shown along the handle to make the graph 3-regular, except at the leaves of the fanin part, which have degree 1. Our desired gadget is formed from the rake by replacing every vertex v by a group of P vertices $v_1, \ldots, v_P$, and replacing every edge $\{u, v\}$ by the $P \times P$ complete bipartite graph between

$u_1, \ldots, u_P$ and $v_1, \ldots, v_P$. In the resulting graph, the $L = P \cdot 2^l$ leaves of the fanin part, each of which has degree P, are the entry vertices of the gadget. All other vertices have degree $3P$. For definiteness we choose $l = 4\lfloor(\log_2 \tau)/4\rfloor$. Thus, the gadget has

$$(2^{l+1} - 1 + \tau - l) \cdot P \leq 3P\tau$$

vertices, and $P\tau/16 < L \leq P\tau$ entry vertices.

In the gadget, assign vertices to layers $0, \ldots, \tau$, counting the entry vertices as layer 0, in the obvious way (left to right in Figure 1). Let σ be a tree of height at most τ with at most P leaves, and all parent-to-child edges labeled from $\{0, \ldots, 3P-1\}$. Then it is not hard to see that there is a labeling of the gadget so that σ can be embedded in it with σ's root at any of its entry vertices. In particular, for each level i of σ (counting the root as level 0), fix a numbering of the vertices on that level from 1 to P (or less). If in σ the root connects to vertex v on level 1 through an edge labeled a ($1 \leq v \leq P$), then in the gadget every vertex in layer 0 will have an edge labeled a to the v^{th} vertex in the (unique) group of P on level 1 to which it is connected. Furthermore, if in σ vertex u on level $i \geq 1$ connects to vertex v on level $i+1$ through an edge labeled a ($1 \leq u, v \leq P$), then in the gadget the u^{th} vertex in each group of P in layer i will have an edge labeled a to the v^{th} vertex in the (unique) group of P on level $i+1$ to which it is connected. All remaining edges of the gadget can be labeled arbitrarily.

We associate with each connecting edge of the graph G_i two integers: a *charge*, initially zero, and a *birthdate*.

Our accounting strategy will be to charge each pebble motion to at most one connecting edge. When an uncommitted edge e in G_i accumulates a charge of τ, we will convert e into a committed edge. More precisely, we will form G_{i+1} from G_i by committing e (as described more fully below). A lower bound on the total running time of M is then the number of committed edges (which will turn out to be $P\lfloor n/49 \rfloor$) times τ.

The initial graph G_0 has one gadget, arbitrarily labeled, $2PL$ uncommitted connecting edges, and associated trees. The start vertex s is an arbitrary vertex in G_0's gadget.

The construction of G_{i+1} from G_i proceeds as follows. Begin with M in its *initial* configuration with all of its P pebbles on s in the current graph G_i. Simulate successive moves of M on G_i until some uncommitted connecting edge accumulates charge τ, where edge charges are determined by the following rules. During a move, suppose M moves pebble p along:

- an edge internal to a gadget or tree. Let e be the connecting edge most recently crossed by p. If e has charge less than τ, then charge the move to e; otherwise there is no charge.
- a connecting edge e (committed or not). Charge the move to the oldest (i.e., least birthdate) connecting edge having charge less than τ. If this is the first step in which a pebble has crossed edge e in either direction, define the birthdate of e to be the current time.

When some uncommitted connecting edge $e = \{u, v\}$ with label $l_{u,v} = a$ accumulates charge τ we stop the simulation, and construct from G_i a new graph G_{i+1} defined as follows.

In the tree Σ_v entered through e remove all vertices and edges not on a path from some pebble to the root. Note that the resulting tree σ_v has at most P leaves. Furthermore, it has external path length at most τ, since each pebble move in it was charged to e. Thus, σ_v can be embedded in a gadget with an appropriate labeling. If possible, choose an entry vertex x of a gadget such that

- x's gadget has a labeling compatible with σ_v (rooted at x),
- x has an uncommitted edge $\{x, y\}$, say with label $l_{x,y} = b$, that has never been crossed by a pebble,
- x and u are not adjacent, and
- x's gadget has remained pebble free since the birthdate of the uncommitted edge e.

(1)

If there is no such x, or if $\{u, v\}$ and $\{x, y\}$ are the *only* uncommitted edges in G_i, then G_{i+1} will have one additional gadget compatible with σ_v, and its associated $2PL$ uncommitted edges and trees. The

vertex x then is chosen to be any one of the new gadget's entry vertices. G_{i+1} is otherwise identical to G_i, except that the trees Σ_v and Σ_y rooted at v and y are removed, and the (uncommitted) edges $e = \{u, v\}$ and $\{x, y\}$ are replaced by the single (committed) edge $\{u, x\}$ with labels $l_{u,x} = a$ and $l_{x,u} = b$.

The behavior of M on G_{i+1} is similar to its behavior on G_i. Suppose in G_i the uncommitted edge e was first crossed during the simulation of the j^{th} move of M (i.e., has birthdate j), and accumulates charge τ during the k^{th}. When M is simulated on G_{i+1}, it will behave exactly as on G_i for the first $j-1$ moves. Between steps j and k those pebbles that crossed edge e in G_i will be in x's gadget in G_{i+1} instead of in σ_v as they were in G_i, but since the gadget contains σ_v as a subgraph rooted at x, their motions in G_{i+1} will exactly reflect their motions in G_i. It is crucial that the chosen gadget was pebble free between steps j and k, so there is no possibility that these pebbles will meet pebbles in G_{i+1} that they did not meet in σ_v in G_i. After step k, each of the $i+1$ committed edges in G_{i+1} will have a charge of τ, and hence M will run for at least $(i+1)\tau$ steps on G_{i+1}.

Continue this process until $G_{P\lfloor n/49 \rfloor}$ is constructed. The final graph G is built from $G_{P\lfloor n/49 \rfloor}$ by

- committing all uncommitted edges that have been crossed by pebbles, as described above,
- deleting all trees,
- joining the remaining uncommitted edges with Δ extra vertices so as to make G have n vertices and be $3P$-regular, and
- designating one of these extra vertices as t.

By an argument similar to one above, M's behavior on G is essentially the same as on $G_{P\lfloor n/49 \rfloor}$. In particular, the edge charges will be the same, so it will run for at least $P\lfloor n/49 \rfloor \tau$ steps without reaching t. We will complete the analysis by showing that $\tau = \Omega((\log n)/\log P)$ and that $\Delta = \Omega(P^2)$, which is sufficient for attaining the correct size and degree of G.

The number of distinct trees σ that can arise is at most $(3P^2)^\tau$, since each can be described by listing the sequence of at most τ moves in the tree, where each move can be specified by giving the edge label followed, and the number of the pebble making the move.

In the graphs G_i and G, there might be many copies of gadgets having each labeling σ. The key claim in establishing the size of G is that there are never more than $P(\tau+3)$ open copies of a gadget with a given label, where an *open* gadget is one having at least one uncommitted edge. In particular, when G_{i+1} is defined, if there are this many open copies of gadgets with labeling compatible with σ_v, then at least one of them will have an entry vertex x satisfying the conditions (1), so a new (open) gadget will *not* be introduced into G_{i+1}. To see this, we show an upper bound on the number of open gadgets that are disqualified from containing x. It is easy to see that at most $2P-1$ entry vertices are adjacent to u in G_i. The more subtle problem is to bound the number of gadgets that can be touched by pebbles between the birth of the uncommitted edge e, and the time at which it has accumulated charge τ. At most P gadgets contain pebbles at the time of e's birth. At most $P-1$ edges older than e can have charge less than τ, because for each such edge f there is at least one pebble that doesn't leave its gadget or tree until f has accumulated charge τ. Each gadget touched by some pebble after the birth of e necessitates the crossing of some connecting edge. Thus after at most $(P-1)\tau$ such crossings, e will be the oldest uncommitted edge, and after at most τ more crossings, e will have charge τ. Thus, at most $P + P\tau$ gadgets can be touched by pebbles during the relevant interval, which establishes the claim.

Now to compute a bound on Δ, note first that $G_{P\lfloor n/49 \rfloor}$ has exactly $P\lfloor n/49 \rfloor$ committed edges, or $2P\lfloor n/49 \rfloor$ instances of a committed edge incident to an entry vertex (called "half-edges" for short). Each closed gadget contributes $2PL$ half-edges, so there can be no more than $2P\lfloor n/49 \rfloor/(2PL) \leq n/(49L)$ closed gadgets in $G_{P\lfloor n/49 \rfloor}$. As argued above, there can be no more than $P(\tau+3)\cdot(3P^2)^\tau$ open gadgets in $G_{P\lfloor n/49 \rfloor}$. Each gadget has size at most $3P\tau$, so

$$n - \Delta \leq \left(\frac{n}{49L} + P(\tau+3)(3P^2)^\tau\right) 3P\tau$$
$$< \frac{48}{49}n + (3P^2)^{\tau+1}\tau(\tau+3).$$

This is less than $n - \Omega(P^2)$ for n sufficiently large, $P \leq n^{1/41}$, and

$$\tau = 9 + 4\left\lfloor \frac{\log n}{8\log(3P^2)} \right\rfloor.$$

Furthermore, recalling that $l = 4\lfloor \log_2 \tau/4 \rfloor$, it is clear that $\tau - l \equiv 1 \pmod 4$, and $\tau - l \geq 5$ since $x - 4\lfloor \log_2 x/4 \rfloor \geq x - \log_2 x \geq 5$ for $x \geq 8$. □

It is interesting to note why the proof would fail if M were allowed to jump pebbles. In the simplified example sketched at the beginning of the proof, we were able to pick an existing gadget in which p must invest τ steps. In the presence of jumping, this fails, since p can always jump out of the new gadget. As a particular foil to the proof above, imagine an automaton that stations one pebble p on an entry vertex of some gadget, and successively moves a second pebble q to each neighbor, jumping q back to p to find the next neighbor. In time $3P \ll \tau$, this has touched all the connecting edges incident to that entry vertex, which was impossible in the construction above.

5 A Lower Bound for the Cycle

In this section we show that nonjumping automata with a constant number Q of states, one active pebble, and a constant number P of passive pebbles are too weak for studying lower bounds on time. In fact, unless $PQ = \Omega(n)$ such automata cannot even traverse all n-vertex cycles, no matter how much time they are allowed. (Proofs are omitted from this abstract.)

Lemma 3 *Let $\alpha \in \{0,1\}^*$. Consider the chain C_α of length $2|\alpha|$ with left endpoint L, right endpoint R, and midpoint M, and edge labels so that α is the labeling from L to M and also from R to M. Then starting at any vertex on C_α that is an even distance from L and traversing according to α terminates at M.*

As a byproduct, there is an interesting corollary concerning universal traversal sequences for the cycle. It is not clear *a priori* that a sequence such as $(00010)^{n^2}$ could not be universal for all cycles. The following corollary shows that this is impossible.

Corollary 4 *For any $\alpha \in \{0,1\}^+$ and any integers n and k, if $|\alpha| < n/2$ then α^k is not a universal traversal sequence for all labeled n-cycles.*

Using similar techniques, Corollary 4 holds for any α such that $\alpha\alpha$ is not a universal traversal sequence for all labeled $(n/2)$-cycles. For instance, it holds for any α of length $O(n^{1.29})$ (Tompa [20]).

Theorem 5 *Any WAG that traverses every labeled n-cycle using Q states, one active pebble, and P passive pebbles satisfies $(P+4)Q \geq n$.*

In contrast, it is easy to see that there is a nonjumping automaton that traverses every labeled n-cycle using a constant number of states and only 2 *active* pebbles, and in addition requires only $O(n)$ time.

Cook and Rackoff [13, Theorem 4.14] presented a family of 3-regular graphs that could not be traversed using a constant number of states and pebbles, even if jumping is allowed. The price paid to capture this strengthened model is a bound that is quantitatively weaker than that of Theorem 5. For instance, they could not rule out the combination $Q = O(1)$ and $P = O(\log\log n)$.

6 Open Problems

The obvious important problem is to strengthen and generalize these lower bounds. The ultimate goal might be to prove that $ST = \Omega(mn)$ for JAGs, or even for general models of computation. At this point, however, it is an open problem to prove $T = \omega(n\log n)$, even for nonjumping automata with only *two* pebbles, one of which is passive, on constant degree graphs, or to prove $T = \omega(n)$ for nonjumping automata with $O(1)$ active pebbles on degree 3 graphs.

It would be interesting to strengthen the result of Section 4 to automata that have more pebbles than the graph's degree. Cook and Rackoff [13, Theorem 4.13] show how to convert lower bounds

on high degree graphs into lower bounds on degree 3 graphs, but unfortunately their conversion seems to rely on the ability to jump.

Acknowledgements

We thank Michael Sipser for showing us the construction generalized in Section 4.

References

[1] 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 *20th Annual Symposium on Foundations of Computer Science*, pages 218–223, San Juan, Puerto Rico, Oct. 1979. IEEE.

[2] N. Alon, Y. Azar, and Y. Ravid. Universal sequences for complete graphs. *Discrete Applied Mathematics*, 27:25–28, 1990.

[3] A. Bar-Noy, A. Borodin, M. Karchmer, N. Linial, and M. Werman. Bounds on universal sequences. *SIAM Journal on Computing*, 18(2):268–277, Apr. 1989.

[4] P. Berman and J. Simon. Lower bounds on graph threading by probabilistic machines. In *24th Annual Symposium on Foundations of Computer Science*, pages 304–311, Tucson, AZ, Nov. 1983. IEEE.

[5] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *19th Annual Symposium on Foundations of Computer Science*, pages 132–142, Ann Arbor, MI, Oct. 1978. IEEE.

[6] M. Blum and W. J. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In *18th Annual Symposium on Foundations of Computer Science*, pages 147–161, Providence, RI, Oct. 1977. IEEE.

[7] B. Bollobás. *Extremal Graph Theory with Emphasis on Probabilistic Methods*, volume 62. CBMS-AMS, 1986.

[8] A. Borodin. Structured vs. general models in computational complexity. *L'Enseignement Mathématique*, XXX:47–65, 1982.

[9] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, June 1989. See also 18(6): 1283, Dec. 1989.

[10] A. Borodin, W. L. Ruzzo, and M. Tompa. Lower bounds on the length of universal traversal sequences. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 562–573, Seattle, WA, May 1989. To appear in *Journal of Computer and System Sciences.*

[11] M. F. Bridgland. Universal traversal sequences for paths and cycles. *Journal of Algorithms*, 8(3):395–404, 1987.

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

[13] S. A. Cook and C. W. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, Aug. 1980.

[14] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.

[15] I. N. Herstein. *Topics in Algebra.* John Wiley & Sons, second edition, 1975.

[16] S. Istrail. Polynomial universal traversing sequences for cycles are constructible. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 491–503, Chicago, IL, May 1988.

[17] H. J. Karloff, R. Paturi, and J. Simon. Universal traversal sequences of length $n^{O(\log n)}$ for cliques. *Information Processing Letters*, 28:241–243, Aug. 1988.

[18] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[19] M. Tompa. Two familiar transitive closure algorithms which admit no polynomial time, sublinear space implementations. *SIAM Journal on Computing*, 11(1):130–137, Feb. 1982.

[20] M. Tompa. Lower bounds on universal traversal sequences for cycles and higher degree graphs. Technical Report 90-07-02, University of Washington, Department of Computer Science and Engineering, July 1990.