Time-Space Tradeoffs for Undirected Graph Traversal

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

Technical Report 93-02-01
February, 1993

An extended abstract of this work appears in the Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, St. Louis, MO, Oct. 1990, pages 429–438.

Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195

Time-Space Tradeoffs for Undirected Graph Traversal *

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

March 2, 1993

Abstract

We prove time-space tradeoffs for traversing undirected graphs, using a variety of structured models that are all variants of Cook and Rackoff's "Jumping Automata for Graphs". Our strongest tradeoff is a quadratic lower bound on the product of time and space for graph traversal. For example, achieving linear time requires linear space, implying that depth-first search is optimal. Since our bound in fact applies to nondeterministic algorithms for *non*connectivity, it also implies that closure under complementation of nondeterministic space-bounded complexity classes is achieved only at the expense of increased time. To demonstrate that these structured models are realistic, we also investigate their power. In addition to admitting well known algorithms such as depth-first search and random walk, we show that one simple variant of this model is nearly as powerful as a Turing machine. Specifically, for general undirected graph problems, it can simulate a Turing machine with only a constant factor increase in space and a polynomial factor increase in time.

*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, CCR-8907960, and CCR-9002891, and by IBM under Research Contract 16980043. A portion of this work was performed while the fourth author was visiting the University of Toronto, whose hospitality is gratefully acknowledged.

†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

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, it is not widely believed that they can be traversed deterministically in polynomial time and small space simultaneously. (See Tompa [35] for a lower bound, and Barnes *et al.* [5] for an upper bound.) In contrast, *undirected* graphs can be traversed in polynomial time and logarithmic space *probabilistically* by using a random walk (Aleliunas *et al.* [2], Borodin *et al.* [13]); this implies similar resource bounds on (nonuniform) deterministic algorithms (Aleliunas *et al.* [2]). More recent work presents uniform deterministic polynomial time algorithms for the undirected case using sublinear space (Barnes and Ruzzo [6]), and even $O(\log^2 n)$ space (Nisan [30]), as well as a deterministic algorithm using $O(\log^{1.5} n)$ space, but more than polynomial time (Nisan *et al.* [31]).

In this paper we concentrate on the undirected case. The simultaneous time and space requirements of the best known algorithms for undirected graph traversal are as 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 [2]. In fact, Broder *et al.* [16] have shown 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 random access machine, obtaining such a tradeoff is beyond the reach of current techniques. Thus it is natural to consider a "structured" model (Borodin [12]), 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 [19]. Such an automaton has a set of states, 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 pebble reaches them. The pebbles represent vertex names that a structured algorithm might record in its workspace. Walking represents replacing a vertex name by some adjacent vertex found in the input. Jumping represents copying a previously recorded vertex name.

Rabin (see [19]), Savitch [33], Blum and Sakoda [9], Blum and Kozen [8], Hemmerling [21] and others have considered similar models; see Hemmerling's monograph for an extensive bibliography (going back over a century) emphasizing results for "labyrinths" — graphs embedded in two- or three-dimensional Euclidean space.

The JAG is a structured model, but not a weak one. In particular, it is general enough to

encompass in a natural way 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 state. Furthermore, as Savitch shows [33], a JAG with the additional power to move a pebble from vertex i to vertex $i+1$ can simulate an arbitrary Turing machine on directed graphs. Even without this extra feature, we will show in Section 3 that JAGs are as powerful as Turing machines for the purposes of solving undirected graph problems (our main focus).

Cook and Rackoff define the time T used by a JAG to be the number of pebble moves, and 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 $\log_2 Q$.) It is well known that st-connectivity for directed graphs can be solved by a deterministic Turing machine in $O(\log^2 n)$ space, by applying Savitch's Theorem [32] to the obvious $O(\log n)$ space nondeterministic algorithm for the problem. Cook and Rackoff show that the same $O(\log^2 n)$ space upper bound holds for deterministic JAGs by direct construction of an $O(\log n)$ pebble, $n^{O(1)}$ state deterministic JAG for directed st-connectivity. More interestingly, they also prove a lower bound of $\Omega(\log^2 n/\log\log n)$ on the space required by JAGs solving this problem, nearly matching the upper bound. Standard techniques ([1, 2]) extend this result to randomized JAGs whose time bound is at most exponential in their space bound. Berman and Simon [7] extend this space lower bound to probabilistic JAGs with even larger time bounds, namely exponential in $\log^{O(1)} n$.

In this paper we use variants of the JAG to study the tradeoff between time and space for the problem of *undirected* graph traversal. The JAG variants we consider are in some ways more restricted than the model introduced by Cook and Rackoff, but in other ways are sometimes more powerful. For example, the variant studied in Section 4 is more restricted in its jumping ability, but is considerably more powerful in another dimension, namely, it is nondeterministic.

Several authors have considered traversal of undirected regular graphs by a JAG with an unlimited number of states but only the minimum number (one) of pebbles, a model better known as a *universal traversal sequence* (Aleliunas *et al.* [2], Alon *et al.* [3], Bar-Noy *et al.* [4], Bridgland [15], Istrail [25], Karloff *et al.* [28]). A result of Borodin, Ruzzo, and Tompa [14] shows 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, perhaps $\Theta(mn)$. The main results of this paper are lower bounds for variants of the JAG that provide progress toward proving this conjecture and, in fact, establish such a lower bound for one variant. These results are outlined below.

The upper bound of $ST = O(m^2 \log^5 n)$ by Broder *et al.* [16] 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 (probabilistically), with the others as fixed landmarks. In Section 4, using essentially the same variant of the JAG, we prove lower bounds of $PT = \Omega(n^2)$ for d-regular graphs ($d \ge 3$), and $PT = \Omega(mn)$ for nonregular graphs, independent of the value of Q, even for nondeterministic

JAGs. For sparse graphs this nearly matches the upper bound. The main difference between the upper and lower bounds is that they are for complementary problems. The upper bound is by a one-sided error probabilistic algorithm for undirected st-connectivity. The lower bound applies to nondeterministic, and hence one-sided error probabilistic, algorithms for st-*non*connectivity. This result does not imply that the algorithm of Broder *et al.* is optimal, but does imply, for example, that it cannot be made both errorless (i.e., zero-sided error) and substantially faster (on this JAG model). It also implies optimality of depth- and breadth-first search, in the following sense. While it is not surprising that linear time is necessary for deciding connectivity (e.g., see Theorem 3), our quadratic lower bound shows the stronger result that achieving linear time requires linear space.

This result also bears on the complexity of undirected st-connectivity, versus that of its complement, st-nonconnectivity. For deterministic, or errorless probabilistic algorithms, of course, the two problems are of equal complexity. For nondeterministic or one-sided error probabilistic algorithms, however, the complexities may differ. In particular, if a problem L is solvable nondeterministically in $O(\log n)$ space, then the complement of L is, too, by the result of Immerman and Szelepcsényi [24, 34]. However, their algorithms are rather slow. For example, a logarithmic space nondeterministic RAM can solve st-connectivity in time $O(n)$, but to solve the complementary st-nonconnectivity problem by the Immerman or Szelepcsényi algorithms requires time $\Omega(n^4)$. Is nonconnectivity intrinsically more difficult? One of our results shows that this is indeed the case, at least on one class of structured models we consider. Namely, although both problems are solvable by a logarithmic space, polynomial time nondeterministic JAG, st-nonconnectivity is provably harder. Specifically, st-connectivity is solvable in $O(n)$ time by a logarithmic space nondeterministic JAG with only one pebble, a constant number of states, and no jumping. In contrast, we show that st-nonconnectivity requires more time, even on a somewhat richer model. Namely, time $\Omega(mn)$ ($\Omega(n^2)$ for regular graphs) is required to solve st-nonconnectivity by a nondeterministic JAG with one movable pebble and any fixed number of unmovable pebbles, even using exponentially many states and jumping.

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; see Section 3. Nevertheless, in Section 5 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 [9], Blum and Kozen [8] and Hemmerling [21]. We will distinguish this nonjumping variant by referring to it as a WAG — "walking automaton for graphs".) More specifically, using a very different and more complex argument, we prove lower bounds on time that are nonlinear in m for a wide range of values of P. In particular, for any deterministic WAG M solving st-connectivity in logarithmic space, there is a family of regular graphs on which M requires time $m^{1+\Omega(1)}$. Near the other extreme, if M uses a number of pebbles that is sublinear in m, there is a family of regular graphs on which M requires time superlinear in m. Although these give the desired quadratic lower bound only at the extreme of linear time, they each at least establish that logarithmic space and linear time are not simultaneously achievable on the nonjumping model when $m = \omega(n)$. (They do not settle the question of simultaneous achievability of logarithmic space and linear time when $m = O(n)$ since the families of regular graphs mentioned above have degree $d = \omega(1)$ and hence

$m = \omega(n)$; see Sections 5 and 7.)

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 [19] investigate JAGs on undirected graphs in this case, showing for example that $PQ = O(1)$ is impossible. For a nonjumping variant of JAGs, in Section 6 we prove the stronger bound $PQ = \Omega(n)$ 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.

Sections 3, 4, 5 and 6 are largely self-contained, and may be read in any order.

2. Graph-Traversing Automata

The problem we will be considering is "undirected st-connectivity": given an undirected graph G and two distinguished vertices s and t, determine if there is a path from s to t.

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 $\lambda_{u,v}, \lambda_{v,u} \in \{0, 1, \ldots, d-1\}$ with the property that, for every pair of distinct edges $\{u, v\}$ and $\{u, w\}$, $\lambda_{u,v} \neq \lambda_{u,w}$. It will sometimes be convenient to treat an undirected edge as a pair of directed *half edges*, each labeled by a single label. For example, the half edge directed from u to v is labeled $\lambda_{u,v}$.

This definition requires that the outgoing labels from each vertex u be distinct. That is, for all u and all neighbors $v \neq v'$ of u we require $\lambda_{u,v} \neq \lambda_{u,v'}$. We will also consider more restricted labelings. We define a graph to be *bijectively labeled* if, in addition, the incoming labels are distinct, i.e., $\lambda_{v,u} \neq \lambda_{v',u}$. A special case of bijective labelings are the *symmetric* labelings, where all edges have the same label in each direction, i.e., $\lambda_{u,v} = \lambda_{v,u}$ for all u, v. (Universal traversal sequences for regular graphs with bijective and symmetric labelings have been considered previously by Hoory and Wigderson [23] and Istrail [26], respectively, although under different names. Both papers used the term "consistent" for these two different classes of restricted labelings.)

Not all graphs have symmetric labelings, and while every graph does have a bijective labeling, such labelings are not known to be computable in logarithmic space. Nevertheless, Lemma 1 below shows that, when considering upper bounds for st-connectivity, there is no loss of generality in restricting attention to symmetrically (and hence, bijectively) labeled graphs. Of course, lower bounds are at least as strong if they also hold when restricted to such graphs.

The reduction mentioned in Lemma 1 is not intended to be implemented on a JAG, but rather on a general model of computation such as a Turing machine.

Lemma 1: There is a simple (e.g., logarithmic space), connectivity-preserving reduction from general labeled graphs to symmetrically labeled graphs. Moreover, the symmetrically labeled graph

has maximum degree at most 3.

Proof: Let $r_{u,v}$ be the rank of $\lambda_{u,v}$ in $\{\lambda_{u,v'} \mid v' \text{ is adjacent to } u\}$. For example, if the graph is regular, $r_{u,v}$ is simply $\lambda_{u,v}$. Replace each vertex of degree d by a d-cycle, if d is even, and by a $(d+1)$-cycle, if d is odd. (For the purposes of this proof, a 2-cycle is simply an edge.) Label these cycles symmetrically using 0 and 1. Replace edge $\{u, v\}$ by an edge from the $r_{u,v}$-th vertex of u's cycle to the $r_{v,u}$-th vertex of v's cycle, symmetrically labeled 2. □

It is not difficult to extend the proof to make the graph in Lemma 1 both symmetrically labeled and 3-regular.

Following Cook and Rackoff [19], a *JAG* is an automaton with Q states and P distinguishable pebbles, where both P and Q may depend on n and d. For the st-connectivity problem, two vertices s and t of its input graph are distinguished. The P pebbles are initially placed on s. Each move of the JAG depends on the current state, which pebbles coincide on vertices, which pebbles are on t, and the edge labels emanating from the pebbled vertices. Based on this information, 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, i must be an edge label emanating from the vertex currently pebbled by p, and p is moved to the other endpoint of the edge with label i; in the latter case, p "jumps" to the vertex occupied by pebble j. (The decision to make t "visible" to the JAG but s "invisible" was made simply to render 1-pebble JAGs on regular graphs equivalent to universal traversal sequences.) A deterministic JAG that determines st-connectivity is required to enter an accepting state if and only if there is a path from s to t. Nondeterministic and probabilistic generalizations of JAGs are defined in the usual way. Note that JAGs are nonuniform models.

There are a number of interesting variants of JAGs. For instance, in Section 4 we will consider a strengthened form of jumping, called "strong jumping," where the automaton's move may also be to select some $v \in \{1, \ldots, n\}$ and jump pebble p to vertex v. On the other hand, in Sections 5 and 6 we will disallow jumping by studying WAGs. 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 5. 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. Of particular interest is the case of one active pebble and $P-1$ passive pebbles, in which case it is natural to think of the automaton itself as the active pebble moving about the graph, picking up and dropping pebbles. This is the model used in Section 6.

Closely related to the passive pebble is the unmovable pebble, which, once placed on the graph, cannot be moved at all. This is the model discussed in Section 4. We will mainly consider unmovable pebbles as a special case of passive pebbles. That is, the automaton starts with a supply of pebbles that are carried and dropped at will (but never picked up). In Section 4.2, however, we will also consider a less uniform placement method where some of the pebbles are placed on the graph before the JAG begins its computation. Detailed definitions of this version are deferred to Section 4.2.

We have defined JAGs running on arbitrary graphs, but our lower bounds generally apply even to JAGs that are only required to operate correctly on regular graphs. The restriction to regular graphs, in addition to strengthening the results, provides comparability to the known results about universal traversal sequences. A technicality that must be considered in the case of regular graphs

is that they do not exist for all choices of degree d and number of vertices n, as is seen from the following proposition.

Proposition 2: d-regular, n vertex graphs exist if and only if dn is even and $d \leq n-1$.

(See [14, Proposition 1], for example, for a proof.) To allow use of Ω-notation in expressing our lower bounds, however, the "time" used by a JAG must be defined for *all* sufficiently large n. To this end, we consider the time used by a JAG on d-regular, n-vertex graphs where dn is odd to be the same as its running time on d-regular, $(n+1)$-vertex graphs. We adopt a similar convention for d-regular symmetrically labeled graphs, which exist if and only if, in addition to the restrictions above, either $d = 0$ or n is even.

It is not difficult to show that st-nonconnectivity requires time $\Omega(m)$ on any of the JAG variants described above, independent of the number of pebbles and states. This result is not surprising, but we will sketch it because of its generality, and also because the proof introduces some ideas we will use subsequently.

Theorem 3: Let n be a multiple of 4, $d < n/2$, and $m = dn/2$. Any JAG, even a nondeterministic one with strong jumping, solving st-nonconnectivity for all symmetrically labeled, d-regular, n-vertex, m-edge graphs requires time $\Omega(m)$ in the worst case.

Proof: With the given constraints on n and d, there is a d-regular, n-vertex, symmetrically labeled graph having its vertices and edges evenly divided between two connected components, one containing s, the other containing t (see [11, Exercise 6.2.1]). Fix a minimal length accepting computation of M on this disconnected graph. Suppose for some $a \in \{0, 1, \ldots, d-1\}$ that pebbles in this computation walk across fewer than $\lfloor m/(2d) \rfloor$ edges labeled a. Then there must be at least one edge labeled a in each component that is not crossed during this computation. These two edges can be cut and rejoined so that the resulting graph is an st-connected graph also accepted by this computation. Hence, M requires at least $m/2$ steps. □

See Theorem 15 for a matching upper bound, which is in fact achieved by a deterministic WAG, even on general graphs.

3. JAGs Have Turing Machine Power

In this section we will show that, although JAGs are structured computational models, they are as "powerful" as Turing machines for the purposes of solving problems about undirected graphs. That is, we will show that any undirected graph problem solvable by a Turing machine is also solvable by a JAG in roughly the same space and time. This holds even on relatively weak variants such as WAGs with one passive and one active pebble. Thus, sufficiently strong lower bounds on JAGs or WAGs will have direct implications for Turing machine complexity.

Since the input conventions for JAGs and Turing machines are quite different, we must specify the correspondence between the two models carefully. For technical reasons, we will focus initially on problems about connected, regular graphs with no distinguished vertices. More general problems, including st-connectivity, will be discussed later. Let $\mathcal{G}$ be the set of all edge-labeled, connected,

regular graphs, where edges are labeled as described in Section 2. A *graph problem* is simply a subset $\mathcal{H} \subseteq \mathcal{G}$. For example, $\mathcal{H}$ might be the set of (connected, regular, edge-labeled) bipartite graphs, or the set of Hamiltonian graphs. To say that a graph problem $\mathcal{H}$ is solvable by a JAG M has the obvious meaning — M, when started with all its pebbles on an arbitrary vertex of $G \in \mathcal{G}$, accepts if and only if $G \in \mathcal{H}$.

Turing machines, of course, work not on graphs, but rather on encodings of graphs. Thus, to say that $\mathcal{H}$ is solvable by a Turing machine M means that M accepts a "reasonable" encoding of a graph $G \in \mathcal{G}$ if and only if $G \in \mathcal{H}$. To be precise, an encoding is *reasonable* if and only if it is interreducible with the "adjacency matrix" representation by a deterministic Turing machine using $O(\log n)$ space. In the adjacency matrix representation, an n-vertex d-regular graph is represented by a string l of n^2 symbols from the alphabet $\{\star, 0, 1, \ldots, d-1\}$. Let $l(i,j)$ denote the $(n \cdot i + j)^{th}$ symbol of l, $0 \leq i, j \leq n-1$. Then $l(i,j) = l(j,i) = \star$ if and only if vertices i and j are not adjacent, and otherwise $l(i,j)$ is the label on the half edge from vertex i to vertex j. Note that reasonable encodings of graphs (at least implicitly) specify a numbering of the vertices, a feature not present in $\mathcal{G}$. Thus, there may be many different encodings of each graph, corresponding to different vertex numberings. M, of course, must accept all or none of these equivalent encodings.

Consider the following "edge list encoding", which will be used throughout this section. The vertex names are distinct, but not necessarily consecutive. An edge is encoded as a triple (i,j,a), where i and j are vertex names and a is the label on the half edge from i to j. The edge list encoding consists of a sequence of such triples, in any order, and possibly with repetitions. It is straightforward to show that this is a reasonable encoding.

The main technical result of this section is that a simple JAG can construct an edge list encoding of its input graph in polynomial time and logarithmic space. This is embodied in Lemma 5 below. One key idea in the proof of Lemma 5 is that a WAG can use a universal traversal sequence (Aleliunas *et al.* [2]) to explore its input. Recall that a universal traversal sequence is guaranteed to visit all *vertices* of a graph. The following simple extension is more useful for our purposes.

A sequence $V \in \{0, 1, \ldots, d-1\}^*$ is said to be a *half edge universal traversal sequence for d-regular, n-vertex graphs* if it has the property that a walk according to V from any start vertex of any d-regular, n-vertex graph G will cross every edge of G at least once in each direction. An analogous definition can be made for nonregular n-vertex graphs of maximum degree d. In this case we define the "walk according to V" so that, at a vertex u of less than maximum degree, the next letter of V selects among u's neighbors evenly. To be precise, when at a vertex u of degree $d(u) = d$, with the next letter of V being $\sigma \in \{0, 1, \ldots, d-1\}^*$, the walk proceeds to the neighbor v of u having $\lambda_{u,v} = \sigma$, just as in the d-regular case. When $d(u) < d$, the walk *remains* at u if $\sigma \geq \lfloor d/d(u) \rfloor d(u)$, and otherwise proceeds to the vertex v having $\lambda_{u,v} = \sigma_i$, where σ_i is the i^{th} smallest label on a half edge leaving u, and $i = \sigma \bmod d(u)$. (A simpler definition of "walk according to V" for nonregular labeled graphs would be to remain at u unless $\sigma = \lambda_{u,v}$ for some v. Under this convention, the bound below would be increased by a factor of $O(n)$.)

Lemma 4: Half edge universal traversal sequences of polynomial length exist for n vertex graphs. In particular, length $O(dn^3 \log n) = O(mn^2 \log n)$ suffices for d-regular graphs, and length $O(m^2 n \log n)$ suffices for all m-edge graphs.

Proof: (Sketch.) The *vertex (half edge) cover time* of a graph G, $C_V(G)$ ($C_E(G)$), is the

maximum, over all vertices u, of the expected number of steps required for a random walk starting at u to reach all vertices (cross all half edges, respectively) of G. The *vertex (half edge) hitting time* of G, $H_V(G)$ ($H_E(G)$), is the maximum, over all pairs u, x, of the expected number of steps required for a random walk starting at vertex u to reach vertex x (respectively, to cross half edge x). Clearly hitting time is never greater than cover time, either for vertices or edges. Let $\mathcal{F}$ be a family of edge-labeled graphs, and define $C_V(\mathcal{F})$ to be the maximum cover time of any graph in $\mathcal{F}$, and similarly for $H_V(\mathcal{F})$. A basic result of Aleliunas *et al.* [2] is that any family $\mathcal{F}$ of d-regular graphs has a (vertex) universal traversal sequence of length $O(C_V(\mathcal{F})\log(n^2|\mathcal{F}|))$. Alon *et al.* [3] and Chandra *et al.* [17] observe that $C_V(\mathcal{F})$ can be replaced by $H_V(\mathcal{F})$ in this expression.

These results extend easily to universal traversal sequences for nonregular graphs of maximum degree d (as defined above) by observing that cover- and hitting times are at most doubled when the random walk is modified so as to remain at a vertex u of degree $d(u)$ with probability $(d - \lfloor d/d(u) \rfloor\, d(u))/d \leq 1/2$. Furthermore, for both regular and nonregular graphs, the technique yields an analogous expression bounding the length of half edge universal traversal sequences, using H_E in place of H_V. Zuckerman [37] observes that $H_E(G) \leq H_V(G) + 2m$ for all graphs G. Aleliunas *et al.* [2] show that $H_V(G) \leq 2m\Delta$, where Δ is the diameter of G. It is well known (cf. Lemma 13) that the diameter of d-regular graphs is $O(n/d)$. The Lemma follows, since there are at most n^{dn} labeled d-regular n-vertex graphs, and at most n^{4m} labeled nonregular m-edge, n-vertex graphs. □

We remark that Lemma 4 implies the same bounds for lengths of vertex universal traversal sequences, asymptotically matching the best known upper bounds for both regular (Aleliunas *et al.* [2], Kahn *et al.* [27]) and nonregular graphs.

The main technical result of this section is the following lemma. For the purposes of this lemma, it is convenient to think of the JAG as a "transducer," i.e., as a machine equipped with a one-way, write-only output tape on which it writes the string encoding the graph given to it as input.

Lemma 5: A deterministic WAG with two pebbles, one of them passive, can construct an edge list encoding of its (connected, regular) input graph in time $n^{O(1)}$ and space $O(\log n)$.

Proof: The idea of the proof is for the WAG to use a universal traversal sequence to systematically explore its input G, generating an encoding of the edges it explores as it goes. The key point is to be able to devise a numbering for the vertices, and to determine a vertex's number when needed.

Call the WAG E, and let s be the vertex on which the pebbles of E start. Suppose G is d-regular, with n vertices. Let V be a half edge universal traversal sequence for d-regular, n-vertex graphs. (Cf. Lemma 4.)

Call E's passive pebble p. Initially, E leaves p on s, then determines the shortest prefix U of V^2 such that $|U| \geq |V|$, and a walk from s according to U ends at s. U has the property that a walk from s according to U returns to s after crossing each edge at least once in each direction. Recall that s is not specially marked in our model. The construction of U allows us to retain s as a landmark without permanently marking it with a pebble.

The vertex number $\#w$ that E assigns to an arbitrary vertex $w \in G$ is the length of the shortest prefix U_w of U such that the walk from s according to U_w ends at w. For instance, $\#s = 0$.

For $1 \leq i \leq |U|+1$, let v_i be the vertex reached from s by walking according to the length $i-1$ prefix of U. Let a_i be the i^{th} symbol of U. Then, for $1 \leq i \leq |U|$, the half edge crossed during the i^{th} step of the walk according to U from s will be the half edge $\{v_i, v_{i+1}\}$, which has label $\lambda_{v_i,v_{i+1}} = a_i$. During the i^{th} phase of the algorithm, $1 \leq i \leq |U|$, E will determine and write onto its output tape the triple $(\#v_i, \#v_{i+1}, a_i)$ defining this labeled half edge.

Suppose at the start of the i^{th} phase that both E and p are on v_i, and that E has stored in its state the values i and $\#v_i$. (Initially, this holds with E and p on $s = v_1$, $i = 1$, and $\#v_1 = 0$.) During the i^{th} phase, E operates as follows.

1. Carry p across the half edge labeled a_i from v_i, then drop p on the vertex reached (v_{i+1}, by definition).
2. Walk from v_{i+1} according to the last $|U| - i$ symbols of U, thus returning to s.
3. Walk from s according to U until p is encountered. The length of this walk is $\#v_{i+1}$.
4. Output the triple $(\#v_i, \#v_{i+1}, a_i)$ defining this labeled half edge.

Note that, at the completion of this process, E is in the configuration desired for the start of phase $i+1$.

The running time of E is $O(|U|^2) = n^{O(1)}$, and E has $n^{O(1)}$ states. □

On bijectively labeled graphs, it suffices to have only one movable pebble.

Lemma 6: A deterministic WAG with one movable pebble, and one arbitrarily placed unmovable pebble can construct a binary string encoding its (connected, regular) bijectively labeled input graph in time $n^{O(1)}$ and space $O(\log n)$.

Proof: The proof is similar to that of Lemma 5. Let U be a half edge universal traversal sequence, and for a vertex w, define $\#w$ to be the length of the shortest prefix of U that walks from w to the fixed pebble. Since the graph is bijectively labeled, these vertex numbers will be unique. (This idea is used by Hoory and Wigderson [23].) For $1 \leq i \leq |U|+1$, let v_i be the vertex reached from the fixed pebble by walking according to the length $i-1$ prefix of U. Suppose again that the values i and $\#v_i$ are stored in the state, the movable pebble is on v_i, and a_i is the i^{th} symbol of U. Then, for the neighbor v_{i+1} of v_i reached via label a_i, $\#v_{i+1}$ can be determined by walking from v_{i+1} to the fixed pebble according to U. After writing $(\#v_i, \#v_{i+1}, a_i)$, the movable pebble can be returned to v_{i+1} by walking from the fixed pebble according to the length i prefix of U. Again, the running time of this algorithm is $O(|U|^2) = n^{O(1)}$. □

Theorem 7 below is the main result of this section. It establishes the equivalence between WAGs and "general machines", which include nonuniform Turing machines as a special case. A *general machine* consists of an input $x_1 x_2 \cdots x_n$ and a set of states. The state set may depend on the length n of the input and, in particular, the number of states may be a function of n. Included in each state is the *input index*, which specifies the index of the next input character to be read. In one move, based on its current state q, input index i, and the input symbol x_i, the machine enters a new state q' with a new input index i', as dictated by a transition function that may also depend

on n. This transition may be deterministic, nondeterministic, or probabilistic, depending on the type of the general machine. Acceptance is defined as it is for the corresponding types of Turing machines. *Time* is defined as the number of moves, and *space* as $\log_2 Q$, where Q is the number of states.

General machines are almost identical to the "recognition machines" defined by Cobham [18], except that recognition machines require the input to be accessed sequentially, whereas general machines allow completely random access to the input. It is also easy to see that Turing machines are a special case, by including the worktape contents and head positions as part of the state of the general machine.

Theorem 7: Let $\mathcal{H}$ be an undirected graph problem as defined above, and let $S(n) = \Omega(\log n)$. $\mathcal{H}$ is solvable using space $O(S(n))$ and time $n^{O(1)} \cdot T(n)$ by a deterministic (nondeterministic, probabilistic) general machine if and only if it is solvable in space $O(S(n))$ and time $n^{O(1)} \cdot T(n)$ by a deterministic (nondeterministic, probabilistic, respectively) JAG. Moreover, the JAG simulating a general machine requires no jumping and only two pebbles, one of them passive. On bijectively labeled graphs, the WAG requires only one movable and one unmovable pebble.

Proof: Let M be a general machine accepting $\mathcal{H}$ in space $S(n)$ and time $T(n)$. By Lemma 5 or 6 above, there is a two pebble deterministic WAG E that can construct a binary string encoding its input graph G. Because of logarithmic space reducibility among reasonable encodings, assume without loss of generality that M operates on the same encoding output by E. We build a WAG W accepting $\mathcal{H}$ by simulating both M and E. Specifically, W maintains M's state as part of its state. If M's input index is i, W simulates E until it generates its i^{th} output bit, and then simulates one step of M (deterministically, nondeterministically, or probabilistically, as appropriate). W continues in this manner until M halts. W's state set must be large enough to encode a state of M and a state of E. This requires $2^{O(S(n))}$ states, or $O(S(n))$ space. Note that it is not necessary to store the string constructed by E; its bits are reconstructed as needed. The simulation by W of each of M's steps requires rerunning the entire computation of E, so W's time bound is $n^{O(1)} \cdot T(n)$.

In the other direction, let J be a JAG that accepts $\mathcal{H}$ using $P(n)$ pebbles $Q(n)$ states, and $T(n)$ time. J is simulated by a general machine M, whose state encodes J's state plus the vertex names on which pebbles currently reside. This requires $Q(n) \cdot n^{P(n)}$ states, or space $P(n)\log_2 n + \log_2 Q(n)$, which is, by definition, J's space bound. M can then simulate a move of J, using its input to determine the vertex name to which a given pebble walks by following a given edge label, which increases the time and number of states by only a polynomial factor. □

Note that a general machine can solve any graph problem in linear space and time (nonuniformly), hence by Theorem 7, a WAG can do so in linear space and polynomial time. Theorems 12 and 15 in Section 4.3 give faster WAG algorithms at this space extreme.

Corollary 8: A JAG solving an undirected graph problem in space $\Omega(\log n)$ can be simulated by a WAG, at the expense of a constant factor loss in space and a polynomial factor loss in time.

Corollary 9: A JAG or WAG solving an undirected graph problem using P pebbles and space $\Omega(\log n)$ can be simulated by one with only two pebbles, at the expense of a polynomial factor loss

in time and a constant factor loss in space (more precisely, a factor of $n^{P+O(1)}$ in the number of states).

The polynomial factor loss in time implicit in Corollaries 8 and 9 is $O(U^2(n))$, where $U(n)$ is the length of a half edge universal traversal sequence (Lemma 4). This factor can be improved to $O(U(n))$ by directly using the proof techniques from Lemmas 5 and 6.

As another illustration of Theorem 7, consider the problem of deciding bipartiteness of a connected graph. It is easy to see that a nondeterministic two pebble WAG can recognize *non*bipartite graphs (guess and verify an odd cycle), but not so easy to see a direct way to recognize bipartite graphs. In fact this is also possible, by the following corollary to Theorem 7 and Immerman and Szelepcsényi's Theorem [24, 34].

Corollary 10: Let $\mathcal{H}$ be an undirected graph problem, and let $S(n) = \Omega(\log n)$. If $\mathcal{H}$ is solvable using space $O(S(n))$ by a nondeterministic JAG or WAG J, then so is its complement $\mathcal{G} - \mathcal{H}$.

Proof: Simulate J by a nondeterministic, $S(n)$ space-bounded general machine M. By a straightforward adaptation of Immerman and Szelepcsényi's Theorem [24, 34], there is a nondeterministic, $S(n)$ space-bounded general machine M' that accepts the complement $\mathcal{G} - \mathcal{H}$. Simulate M' by a nondeterministic, $S(n)$ space-bounded WAG. □

We know no substantially simpler method for recognizing bipartite graphs. Implementation of the Immerman/Szelepcsényi method on a JAG seems to require construction of a vertex numbering, which is the key idea in Lemmas 5 and 6.

Algorithmic problems about graphs often have input parameters other than the graph itself. For example, consider the shortest path problem: given a connected undirected graph G, two designated vertices s and t in G, and an integer k, is there a path from s to t of length at most k? The results above are easily extended to encompass such problems by incorporating integers such as k into the WAG's initial state, and marking "designated" vertices or edges with pebbles, or making them "visible" to the WAG as we did for st-connectivity. Thus, for example, the shortest path problem is solvable in (deterministic) logarithmic space by a WAG if and only it is so solvable by a general machine. This problem is of particular interest since it is a problem about undirected graphs that is known to be complete for *non*deterministic logarithmic space. Hence, it is plausible that complexity results for WAGs will solve a long-standing open problem in Turing machine complexity.

Finally, we mention that the restriction to regular graphs in the above results is only a technicality. Lemmas 5 and 6 are modified easily to accommodate nonregular graphs, since by Lemma 4 there are universal traversal sequences of polynomial length for such graphs. The restriction to connected graphs is only slightly more problematic. Obviously a WAG with only one active pebble cannot explore more of its input graph than the connected component initially holding that pebble. With strong jumping, or with an active pebble in each connected component, or some other mechanism for accessing all components, the results could be extended easily to nonconnected graphs.

4. Unmovable Pebbles

A plausible paradigm for an *st*-connectivity algorithm is to choose and mark a small number of "landmark" vertices in the graph, based perhaps on local properties like proximity to low or high degree vertices or certain small subgraphs, then to explore the graph with these landmarks fixed. This paradigm motivates our study of JAGs with unmovable pebbles.

Depth- and breadth-first search are examples of algorithms where vertices are permanently marked. The undirected *st*-connectivity algorithm of Broder *et al.* [16] is a more complex example of this paradigm. In outline it operates as follows. First, s and t are marked by pebbles. Then $P-3$ other pebbles are placed on the graph at random. More precisely, 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. For example, if the algorithm has learned that pebbles 1 and 2 are in the same connected component, and similarly for pebbles 3 and 4, and during a walk from pebble 1 the algorithm reaches pebble 4, then it can infer that all four pebbles are in the same component. Broder *et al.* show that, in time $O(m^2 \log^5 n/S) = O(m^2 \log^4 n/P)$, 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 essentially a probabilistic JAG, except that the unmovable pebbles are "preplaced" probabilistically without walking to their locations. On a regular graph, Broder *et al.*'s algorithm could be implemented by a probabilistic JAG with strong jumping. On nonregular graphs, the model would have to be extended to allow the dependence of pebble preplacement on vertex degree. In Section 4.2 we will discuss and prove a lower bound for such a model that allows preplacement of pebbles. Section 4.1 first gives the lower bound for the simpler basic model, i.e., without preplacement. Section 4.3 shows that this lower bound is tight for the model we consider.

Note that these lower bounds apply to models that are sufficiently rich to admit depth- and breadth-first search, and the algorithm of Broder *et al.* Thus, as corollaries we establish three facts claimed in the introduction — that depth-first search is space-optimal among linear time algorithms, that Broder *et al.*'s algorithm cannot be made both errorless and substantially faster, and that closure under complementation is intrinsically slow (all with respect to this class of models, of course).

4.1. A Lower Bound for Unmovable Pebbles

In this section, we prove an $\Omega(n^2/P)$ lower bound on the time for a nondeterministic P-pebble JAG to solve *st*-nonconnectivity. We first prove a basic lower bound for regular graphs of degree $d = 3$. Several generalizations are sketched later.

Theorem 11: Let M be any nondeterministic JAG with strong jumping that has 1 active pebble and $P-1$ unmovable pebbles. If M determines *st*-nonconnectivity for all 3-regular symmetrically labeled graphs, then M requires time $\Omega(n^2/P)$.

Proof: The proof generalizes the main lower bound technique introduced by Borodin *et al.* [14]. Assume without loss of generality that n is a multiple of 4. (If not, set aside 6 vertices in a 3-regular connected component containing neither s nor t.) 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 an edge, called a "rung," to the corresponding vertex on the other cycle. Fix any numbering of the vertices and any symmetric 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.* [14], 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 $\{u^b, v^{b \oplus x_i}\}, b \in \{0,1\}$, with labeling $\lambda_{u^b, v^{b\oplus x_i}} = \lambda_{u,v} = \lambda_{v,u} = \lambda_{v^{b\oplus x_i}, u^b}$, where $\oplus$ denotes the EXCLUSIVE OR operation. 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}, and that all graphs in $\mathcal{G}$ are symmetrically labeled.

The key property of the family $\mathcal{G}$ of graphs is that a walk on G_x is identical to such a walk on G_{0^r}, 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 exactly when 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 $1 \leq i \leq r$, and 0 for all unswitchable edges.

Intuitively, M gains information about connectivity only by *walking* to a pebble; nothing is learned (directly) about the existence or nonexistence of a path from u to v by jumping from u to v. We exploit this fact, together with the fact that pebbles on average are far apart, to argue that M must execute many walking steps.

Note that s and t are *not* connected in G_{0^r}, hence M must have at least one accepting computation on G_{0^r}. Fix one such computation γ of minimal length. Assume that two extra unmovable pebbles are placed on the distinguished vertices s and t. Now in G_{0^r} "mark" both copies of each vertex that received an unmovable pebble during the computation γ. Break γ into sequences of walk moves that (1) begin with a walk move from a vertex that either is marked or was the target of a jump in the immediately preceding step, and (2) end with the next walk move into a vertex that either is marked, is the source of a jump move in the immediately following step, or is the last move of γ. Discard any such sequence that does not end at a marked vertex. Suppose there are w sequences remaining. Each of these sequences naturally corresponds to a connected sequence of edges in G_{0^r}. Notice that if, for some x, one of the w sequences is of odd parity with respect to x, then the computations of M on G_{0^r} and on G_x may diverge at the end of this sequence, since a pebble encountered on one may not be encountered on the other. This cannot occur if all sequences have even parity with respect to x:

Claim: For every $x \in \{0,1\}^r$, if each of these w edge sequences is of even parity with respect to x, then γ is also an accepting computation for G_x.

To see this, we show by an induction on i, that after making the moves dictated by γ up to the end of the i^{th} sequence (including the discarded sequences), the configurations of M on G_{0^r} and on G_x are identical, with the exception that the movable pebble will be on opposite copies of a vertex if the net parity with respect to x of the i^{th} sequence is odd. (This can happen only if this is a discarded sequence.) Basically, this is true since all the "interesting" events in the computation γ, i.e., dropping or encountering pebbles, occur at marked vertices, and we've taken care that all walks between these interesting points are of even parity in G_x just as they were in G_{0^r}. The base case ($i = 0$) is vacuous. For the induction step, first note that the configurations at the start of the i^{th} sequence are the same on both graphs, since if they differed at the end of the $(i-1)^{st}$, then all intervening steps were jumps. All steps within the i^{th} sequence are walk steps into unmarked vertices, hence no pebbles are encountered during those steps in either G_{0^r} or G_x. Since both graphs are 3-regular, all unpebbled vertices "look alike", so the i^{th} sequence of walk moves of γ in G_{0^r} is also a legal sequence of moves in G_x, and carries the movable pebble to the same place in both graphs, up to the parity of the sequence with respect to x. This completes the proof of the claim.

As noted earlier, G_x is connected for all $x \neq 0^r$, hence must not be accepted by M. Thus it must be that there is no $x \neq 0^r$ for which the w sequences all have even parity. Equivalently, it must be that the corresponding homogeneous system of w linear equations in r unknowns over $GF(2)$ has no nonzero solution.

Let S be the set of r switchable edges. For each $e \in S$, let $\mathrm{dist}(e)$ be the distance from e to the closest marked vertex, where the distance from an edge to a vertex is defined to be the length of a shortest path containing both. Let m be the maximum integer such that some switchable edge e has $\mathrm{dist}(e) = m$. For any nonnegative integer d, let $S_d = \{e \in S \mid \mathrm{dist}(e) \geq d\}$, and let r_d be the number of switchable edges e with $\mathrm{dist}(e) = d$, so that $r_d = |S_d| - |S_{d+1}|$.

Now it must be the case that, for all $d \leq m$, at least $|S_d|$ of the w walks each have length at least d. If this were not the case, then the edges in S_d would appear collectively on fewer than $|S_d|$ walks or, equivalently, the variables corresponding to these edges would occur in fewer than $|S_d|$ of the homogeneous equations. Set the variables corresponding to the other $r - |S_d|$ switchable edges to 0, and these $|S_d|$ to some nonzero solution, which must exist in a homogeneous system with fewer equations than unknowns (Herstein [22, Corollary to Theorem 4.3.3]). Since such a nonzero solution cannot occur, we have a contradiction.

Thus, at least r_m of the w walks each have length at least m, an additional r_{m-1} each have length at least $m-1$, etc. In other words, M makes at least

$$\sum_{d=1}^{m} d \cdot r_d = \sum_{e \in S} \mathrm{dist}(e)$$

moves. This last sum is minimized when the $O(P)$ marks are equidistantly distributed around the cycle, in which case the sum is $\Omega(rn/P) = \Omega(n^2/P)$. □

Using Hall's Theorem [20], one can in fact prove somewhat more about the w walks: each switchable edge in S can be assigned a unique walk that contains it.

Next, we will sketch several promised generalizations to the theorem. First, to extend the result to d-regular graphs, $d \geq 3$, we generalize the squirrel cage graph H. Note that K_{d-1}, the $d-1$

vertex complete graph, is $d-2$ regular. Form the new d-regular, $n/2$ vertex graph H from $(d-1)$ cycles of length $n/(2(d-1))$ by joining corresponding groups of $(d-1)$ vertices as K_{d-1}. (An extra gadget is needed if $2(d-1)$ does not divide n.) The rest of the argument is essentially as before, except that there are more switchable edges (all but a spanning tree of H, hence $\Theta(dn)$ of them), but on average they are closer to marked vertices ($\Omega(n/(dP))$ average distance). The result is still an $\Omega(n^2/P)$ lower bound, independent of d. To provide some intuition of why the bound does not increase with d, note that any connected d-regular graph has diameter $O(n/d)$, a corollary of Lemma 13 below.

A better bound is possible for nonregular graphs. For n vertex graphs of maximum degree d, one can prove an $\Omega(dn^2/P)$ lower bound, provided $n/(4d) \geq 2P$. Again, the key point is to choose H appropriately. In this case it suffices to take H to be an $n/4$ vertex cycle, attached at evenly spaced intervals to $n/(4d)$ copies of K_d. Most of the $\Theta(dn)$ edges are switchable, and their average distance from any placement of P pebbles is $\Omega(n/P)$. In Section 4.3 we prove matching upper bounds for both the regular and nonregular cases, demonstrating that this disparity in bounds is inherent in the problem.

The remaining generalization promised above is to the case where the automaton can move the "unmovable" pebbles a limited number of times. (A detail about the algorithm of Broder *et al.* that we oversimplified above is that it rerandomizes the placement of the $P-3$ landmark pebbles $O(\log n)$ times.) Suppose M is a P-pebble JAG of this more general form. Suppose pebbles are placed on at most P' vertices during M's computation. Then a straightforward adaptation of the proof of Theorem 11 shows that M requires time $\Omega(n^2/P')$. Note that, as long as the number of pebble placements is sublinear, the time must be superlinear. However, any graph in the family $\mathcal{G}$ built from squirrel cage graphs as above can be traversed in linear time by a deterministic 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 Section 5.

4.2. Preplacement of Unmovable Pebbles

As we have noted earlier, the JAG is a powerful yet restricted model. It is conceivable that there is certain useful information about graphs that is intuitively "easy to compute," yet hard for JAGs to compute. That is, there might be certain information about an input graph G that (1) could be collected easily by a more flexible computational device such as a logarithmic space RAM, that (2) would greatly facilitate a JAG's determination of the st-nonconnectivity of G, yet (3) is difficult or impossible for a JAG to collect. If this were the case, it might "explain" (and trivialize) the strong lower bound given in the previous section.

The algorithm of Broder *et al.* again furnishes a motivating example. Recall that their initial (random) pebble placement is dependent on vertex degree. On nonregular graphs, a JAG cannot duplicate this behavior without visiting all vertices, a slow or even impossible process for, say, a probabilistic JAG without strong jumping. Yet this is an easy process for a RAM, and a crucial one for the efficiency of their algorithm. (Note that the rest of their algorithm *can* be performed efficiently by a JAG.) Generalizing this slightly, it might be useful to know how many neighbors each vertex has at distance two. Although this information is easily computed by a RAM, as far as we know it is not easily computable by a JAG with one movable pebble and a limited number

of unmovable pebbles, even a nondeterministic one with strong jumping.

Does our lower bound rest on this or similar deficiencies of the JAG model? In this section we give evidence that it does not. We generalize the model to allow precomputation on the input and preplacement of (unmovable) pebbles, and show that a similar lower bound holds. 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 as described below.

Let $N_\rho(G)$ denote a list $G_1, G_2, \ldots G_n$ of edge labeled 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 for a triangle free graph, and ignoring edge labels, $N_1(G)$ 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 $U \in \{1, 2, \ldots, n\}^{P'}$. Given G, the P' unmovable pebbles are placed on the sequence of vertices $f(N_\rho(G))$, and then M is run on the resulting pebbled graph. The definition can be further generalized to allow f to select M's initial state. Additionally, it can be generalized in a straightforward way to probabilistic or nondeterministic precomputation by letting f be a relation, and selecting a value from its range probabilistically or nondeterministically. For instance, the algorithm of Broder *et al.* can be executed by a probabilistic JAG with probabilistic 1-precomputation.

The proof of Theorem 11 immediately extends to show an $\Omega(n^2/P)$ lower bound on nondeterministic JAGs with nondeterministic 1-precomputation. The only changes needed in the proof are to note that the initial pebble placement and state $f(N_1(G_{0^r}))$ are considered to be part of the fixed accepting computation γ, and to note that all graphs in $\mathcal{G}$ are 3-regular, symmetrically labeled, and triangle-free, hence $N_1(G_{0^r}) = N_1(G_x)$ for all $x \in \{0,1\}^r$, and so this initial configuration is also legal in G_x.

As a concrete example of the potential utility of precomputation, we note that the squirrel cage family $\mathcal{G}$ defined above can be traversed quickly, provided the unmovable pebbles can be placed based on vertex neighborhoods of radius two, generalizing Broder *et al.*'s use of vertex degree. Specifically, in G_x, a vertex v will have 4, 5, or 6 distinct neighbors at distance two depending on whether the "rung" of the squirrel cage incident to v is of the same parity as both, one, or neither, respectively, of the two nearest nonincident rungs. Thus, 2-precomputation *alone* suffices to distinguish the disconnected graph G_{0^r} (every vertex has 4 neighbors at distance two) from all the connected members of $\mathcal{G}$ (some vertex has more than 4 neighbors). Furthermore, by placing one unmovable pebble on any vertex with more than 4 neighbors at distance two, a WAG with no additional pebbles can traverse the entire graph in linear time.

However, we can show that radius two, or indeed any constant radius, does not help in general. That is, we can further generalize the proof of Theorem 11 to use families of graphs in which switched edges do not alter the local structure within any fixed radius ρ. This is done by choosing a d-regular bipartite graph R whose girth is at least $2\rho+2$ and whose size $|R|$ is $d^{O(\rho)}$ (Bollobás [10, Chapter 3]), and then constructing the half-size graph H by connecting $c = \lfloor n/(2|R|) \rfloor$ copies of R in a cycle. One way to do this is to choose a fixed edge $\{u, v\}$ in R, remove this edge from each copy of R, then insert an edge from u in the i^{th} copy of R to v in copy $(i+1) \bmod c$, $0 \le i < c$. Note that, for every cycle in G_x, there is a corresponding cycle in G_{0^r} that is no longer, so all graphs in $\mathcal{G}$ have

girth at least $2\rho+2$. Furthermore, note by Hall's Theorem [20] that R can be symmetrically labeled since it is regular and bipartite, hence so can G_{0^r}. The key new idea in the proof is that the list $N_\rho(G)$ of radius ρ neighborhoods of any symmetrically labeled, d-regular, girth $2\rho+2$ graph G will simply consist of n identical symmetrically labeled, degree d, complete trees of height ρ. Thus, ρ-precomputation cannot distinguish between G_{0^r} and G_x. The remainder of the proof is essentially unchanged. Thus, nondeterministic JAGs with nondeterministic ρ-precomputation require time $n^2/(d^{O(\rho)}P)$ to solve st-nonconnectivity for d-regular graphs.

We remark in closing this section that ρ-precomputation seems to be orthogonal to pebble placement by the JAG itself. For instance, as noted above, deterministic 2-precomputation may be helpful even to a nondeterministic JAG with strong jumping on as simple a family as the basic squirrel cage family. On the other hand, there are cases where even a weak model such as a deterministic WAG can place pebbles more effectively than can be done by deterministic precomputation. Specifically, we again consider the simple 3-regular squirrel cage family, but enlarged to include all $n!$ permutations of vertex labels for each G_x, $x \in \{0,1\}^r$. Suppose all unmovable pebbles are placed by deterministic 1-precomputation. Then an $\Omega(n^2)$ lower bound applies for $P \leq n - \Omega(n)$, since all pebbled vertices might be concentrated on one part of the squirrel cage pair. On the other hand, a deterministic WAG (knowing the edge labeling) can easily walk one of the cycles, dropping its $P-1$ unmovable pebbles at evenly spaced positions around the cycle. It is then a simple matter to test the switchable edges one after the other from the nearest pebble, hence solving st-connectivity in time $O(n^2/P)$.

4.3. An Upper Bound for Unmovable Pebbles

A natural question to ask is whether the lower bounds given in Section 4.1 can be improved. Recall that Theorem 11 shows time $\Omega(mn/P)$ ($\Omega(n^2/P)$ for regular graphs) is required by JAGs with unmovable pebbles and strong jumping, even with an unbounded number of states. We will close this section by showing that this bound cannot be improved: on the model to which the lower bounds apply, exploiting an unbounded number of states we give matching upper bounds on time for a given number of pebbles, even without jumping. More strongly, we show that *any* graph problem, as defined in Section 3, can be solved within the same bounds.

Theorem 12: Let $\mathcal{G}$ be the set of all bijectively labeled graphs (all bijectively labeled regular graphs). For any $P \geq 2$, the following sets can be recognized by a nondeterministic WAG with 1 movable pebble, $P-1$ unmovable pebbles, an unbounded number of states, and time $O((mn/P)+m)$ ($O((n^2/P)+m)$ in the case of regular graphs):

1. the set of st-nonconnected graphs in $\mathcal{G}$, or
2. any set $\mathcal{H}$ of connected graphs in $\mathcal{G}$.

The main import of this result is to show the limits of the proof technique used in Theorem 11. For example, we do not believe that st-nonconnectivity can be solved by a nondeterministic JAG in time $O(mn)$ and space $O(\log n)$ simultaneously. The fastest known logarithmic space nondeterministic JAG for st-nonconnectivity is much slower than this. Indeed, no better method is known than to use a universal traversal sequence, i.e., a deterministic one pebble WAG, which may require

time $\Omega(m^2 n \log n)$ for nonregular graphs (Lemma 4). However, Theorem 12 shows that to obtain a lower bound greater than that of Theorem 11 we must somehow exploit a bound on the number of states, as well as the number of pebbles. To date, the only lower bounds involving number of states are the relatively weak ones of Cook and Rackoff [19] and of our Section 6. (It might also be possible to exploit nonbijective labelings but, in light of Lemma 1 and the remarks following the proof of the theorem, this issue is a technicality of the model that is not of fundamental importance to the computational complexity of *st*-connectivity.)

The following facts are needed in the proof of Theorem 12.

Lemma 13: Let G be a connected d-regular graph, u and v be any two vertices in G, and $\mathrm{dist}(u,v) = l$ be the length of a shortest path between them. Then there are at least $(d+1)\lfloor (l+2)/3 \rfloor$ vertices in G within distance l of u.

Proof: Let $\Gamma(x) = \{\, y \mid \mathrm{dist}(x,y) \leq 1 \,\}$. Fix a shortest path $u = u_0, u_1, \ldots, u_l = v$ from u to v. Then $\Gamma(u_0), \Gamma(u_3), \ldots, \Gamma(u_{3\lfloor (l-1)/3 \rfloor})$ are pairwise disjoint, for otherwise there would be a shorter path from u to v. Furthermore, these sets are all of size $(d+1)$, and all are within distance l of u. □

Corollary 14: Let G be a connected d-regular graph, and s a vertex in G. For any positive integer $P \leq n/d$, there exists a set S with $s \in S$ and $|S| \leq P$ such that every vertex of G is within distance $l = 2\lceil 1 + 3n/((d+1)P) \rceil = O(n/(dP))$ of some member of S.

Proof: Construct $S = \{s_0, s_1, \ldots\}$, where $s_0 = s$ and, for $i \geq 1$, s_i is chosen to be any vertex at distance greater than l from $\{s_0, \ldots, s_{i-1}\}$. The neighborhoods of radius $l/2$ around the s_i's are pairwise disjoint. Furthermore, by Lemma 13, each of these neighborhoods will be of size at least $(d+1)\lfloor (l/2+2)/3 \rfloor \geq n/P$. Hence at most P members of S can be chosen before no vertices of G remain at distance greater than l. □

The analogous results for nonregular graphs are that at least $(l+1)$ vertices are within distance l of u, hence P vertices can be chosen so that every vertex is within distance $2n/P$ of a chosen vertex. The proofs are similar, but easier.

Finally, we prove the theorem.

Proof (of Theorem 12): The approach is to nondeterministically guess the graph, then verify the guess. First we prove part 1: we describe a nondeterministic WAG M accepting *st*-nonconnected graphs.

Let G be the input graph. It suffices to verify that the connected component C of G containing s does not contain t. Let $l = 2n/(P-1)$, or $l = 2\lceil 1 + 3n/((d+1)(P-1)) \rceil$ in the case of regular graphs. By Corollary 14, for any n-vertex graph G and designated vertex s, there is a set of $P-1$ vertices including s such that every vertex of C is within distance l of a member of this set. Leave one unmovable pebble on s (the initial location of the movable pebble), and place the other $P-2$ unmovable pebbles on arbitrary, distinct vertices selected nondeterministically during a walk γ of length at most $2(n-1)$ from s back to s. (This walk is long enough to traverse a spanning tree of C, hence any vertex may be pebbled.)

M proceeds by guessing and recording in its state an $n' < n$ vertex, connected, bijectively labeled graph B with $P-1$ distinct vertices marked by numbered pebbles. M then verifies that there is a surjective homomorphism ϕ from B to C. That is, ϕ is a surjection preserving pebble placement, vertex degree, adjacency, and edge labeling. Thus, for all vertices u in B, (1) there is a pebble p on $\phi(u)$ in C if pebble p is on u in B, (2) degree(u) = degree$(\phi(u))$, and (3) for all edges $\{u,v\}$ in B, if $\lambda_{u,v} = a$ then $\lambda_{\phi(u),\phi(v)} = a$. (It might seem more natural to guess an *iso*morphic graph B, and it would not be difficult to modify M to do this, but a homomorphism suffices and is easier to verify.) To complete the algorithm, M visits $\phi(v)$ in C for all $v \in B$, accepting if and only if none is the specially marked vertex t. (Recall that M can sense when it has a pebble on t.)

We now show how to construct and verify the homomorphism ϕ. A key property of a bijectively labeled graph, used earlier in Lemma 6, is that for any sequence σ of edge labels, and any vertices u, u' and v, if walks following σ from both u and u' end at v, then $u = u'$. Otherwise, the graph is nonbijectively labeled at the vertex where the two paths last converge. This property is central to constructing and verifying the homomorphism. In particular, recall that a JAG has no access to vertex numbers of the input graph. Instead, we will identify vertices by paths to or from pebbles.

M now runs a breadth-first search of B, with the queue initially containing all the pebbled vertices. The result is a spanning forest of B with B's pebbles as the roots. Reject if any tree has height greater than l. Otherwise, for all vertices u in B, let $\rho(u)$ be the number of the pebble marking the root of the tree containing u, let $\sigma(u)$ be the sequence of edge labels on the unique path of tree edges from $\rho(u)$ to u, and let $\sigma^{-1}(u)$ be the sequence of labels in the reverse direction, i.e., from u to $\rho(u)$. For all vertices u in B, define $\phi(u)$ to be the vertex reached in C by walking from the vertex marked by pebble $\rho(u)$ according to the sequence $\sigma(u)$.

M now performs the following test.

For all edges $\{u,v\}$ in B, say with labels $\lambda_{u,v} = a$ and $\lambda_{v,u} = b$, M verifies that in C the walk $\sigma(u)a\sigma^{-1}(v)$ ends at $\rho(v)$ when started from $\rho(u)$, and that $\sigma(v)b\sigma^{-1}(u)$ returns to $\rho(u)$ from $\rho(v)$. During this process, at the first visit to $\phi(u)$ for each u in B, M also verifies that degree(u) = degree$(\phi(u))$ (with the same set of labels).

We now show that ϕ is a surjective homomorphism if and only if this test succeeds. First, suppose ϕ is a surjective homomorphism. For any vertices x and y in B, if a walk from x according to α ends at y, then a walk from $\phi(x)$ in C according to α must end at $\phi(y)$. This is shown easily by induction on the length of α, using the fact that $\lambda_{u,v} = \lambda_{\phi(u),\phi(v)}$ for all edges $\{u,v\}$. By construction, for any edge $\{u,v\}$ in B, the walk in B from $\rho(u)$ according to $\sigma(u)\lambda_{u,v}\sigma^{-1}(v)$ must end at $\rho(v)$. Furthermore, by construction, if vertex w in B holds a pebble, then $\phi(w)$ holds the same pebble in C. Consequently, each of M's "walk" tests will succeed. By the assumption that ϕ is a homomorphism, each of M's degree tests will also succeed, and so ϕ passes the test.

Conversely, suppose the test succeeds. We argue that that ϕ is a surjective homomorphism. Note that by construction a walk from $\rho(v)$ according to $\sigma(v)$ ends at $\phi(v)$, for all v. We claim first that the reverse also holds: a walk from $\phi(v)$ according to $\sigma^{-1}(v)$ ends at $\rho(v)$, for all v. If v has a pebble, this is trivial. Otherwise v is the child of some u in the spanning forest of B. Then $\rho(v) = \rho(u)$ and $\sigma(v) = \sigma(u)a$ for some a. Since the test succeeds, $\sigma(u)a\sigma^{-1}(v)$ goes from $\rho(v)$ to $\rho(v)$. But the first part $\sigma(u)a$ goes from $\rho(v)$ to $\phi(v)$, so the last part $\sigma^{-1}(v)$ must go from $\phi(v)$ to $\rho(v)$, establishing the claim. Note that as a consequence, if a walk in C from a vertex w according to $\sigma^{-1}(v)$ ends at $\rho(v)$, then $w = \phi(v)$, since C is bijectively labeled.

The following properties of ϕ are now easily established.

1. For all pebbled vertices u in B, $\phi(u)$ holds the same pebble. This holds by construction.

2. For all $u \in B$, degree(u) = degree$(\phi(u))$. This holds since M explicitly tests for this condition, and by assumption the test succeeds.

3. For all adjacent vertices $u, v \in B$, $\phi(u)$ and $\phi(v)$ are adjacent, with $\lambda_{\phi(u),\phi(v)} = \lambda_{u,v}$ (and $\lambda_{\phi(v),\phi(u)} = \lambda_{v,u}$). This holds since $\sigma(u)\lambda_{u,v}\sigma^{-1}(v)$ walks from $\rho(u)$ to $\rho(v)$, by construction the vertex reached by $\sigma(u)$ is $\phi(u)$, and by the remark above the vertex from which $\sigma^{-1}(v)$ reaches $\rho(v)$ is $\phi(v)$.

4. ϕ is surjective. If this did not hold, there would be a vertex in C *not* in the range of ϕ that is adjacent to a vertex $\phi(u)$ that *is* in the range of ϕ. However, this is impossible, since by property 2, degree(u) = degree$(\phi(u))$, and by property 3, $\phi(u)$ has degree(u) neighbors that *are* in the range of ϕ.

Thus, ϕ is a surjective homomorphism, as claimed.

If M accepts, then there is an accepting computation in which B is isomorphic to C, hence has at most m edges. In this computation, the algorithm makes $O(m)$ walks, each of length $O(l)$, hence the total running time is $O(ml)$, as desired. Note that M can move between the pebbles in C by walking γ, the tour used initially to drop the pebbles, which adds only $O(n)$ to the time.

The proof of part 2 of the theorem is similar. The main difference is that the graph B guessed by M will have n vertices, rather than $n' < n$. M then verifies that this graph is isomorphic to the input graph G, accepting (nonuniformly) if and only if it is in $\mathcal{H}$. Note that the homomorphism test given above suffices to verify that B is isomorphic to G, since they have the same number of vertices. □

As in Section 3, these results can be generalized to graph problems with other input parameters, and/or to other problems about unconnected graphs, given an appropriate mechanism for accessing all connected components.

The restriction of Theorem 12 to bijectively labeled graphs can be relaxed at the expense of adding one passive pebble, as follows. The constructions of $\phi(u), \sigma^{-1}(u), \sigma(u)$, and $\rho(u)$ are as before. With a nonbijectively labeled graph it remains true that a walk from $\phi(u)$ according to $\sigma^{-1}(u)$ will end at $\rho(u)$, but it is no longer true that $\phi(u)$ is the only vertex with this property. To verify that the active pebble is on vertex $\phi(u)$, we instead leave the passive pebble there, then verify that $\sigma^{-1}(u)$ walks to $\rho(u)$, from which $\sigma(u)$ returns to the passive pebble. The remainder of the algorithm is unchanged.

As a final observation, the following theorem shows that, at the extreme where $P = n$, the WAG of Theorem 12 can be made deterministic.

Theorem 15: The set of *st*-nonconnected graphs, and arbitrary sets of connected graphs (nonregular, under general labelings) can be recognized in time $O(m)$ by a deterministic WAG with one active pebble and n unmovable pebbles.

Proof: Rather than guessing the input graph, as in Theorem 12, the WAG simply does a systematic traversal of it, akin to a depth-first search, placing a pebble on each vertex. With jumping, or with symmetric edge labels, depth-first search itself would be easy to implement, but lacking both it seems difficult to quickly return after crossing a "back edge" whose reverse label is unknown. We avoid this problem with the following algorithm, which is also akin to an algorithm for finding Euler tours.

M places a distinctly labeled pebble on each vertex it visits, thus effectively numbering the vertices. M records in its state the source, destination, and label of each half edge it crosses. It will eventually cross each half edge, so at termination it will have in its state a complete description of the graph. Connectivity or other properties of the graph can then be determined directly (nonuniformly).

M starts at s, initializing a stack in its state to contain s. At a general step, when at a vertex u with u on top of the stack, if there is a previously uncrossed half edge leaving u, say (u, v), then M crosses this edge, pushing v onto the stack. (M pebbles v if it does not already hold a pebble.) If there are no previously uncrossed half edges leaving u, then M backtracks by popping u from the stack, and returning to v, where v is the new top of stack. By assumption M has previously crossed the (u, v) half edge, and so knows its label. In either case, the process is repeated at v. M terminates when the stack is emptied.

It is easy to see that every visited vertex is pushed onto the stack, and none is removed from the stack until all its outgoing half edges have been traversed. Thus, M will visit all vertices reachable from s. M's running time will be exactly $4m$, since exactly two moves can be charged to each half edge (u, v) — one for the first move by M across that half edge, when v is pushed (on top of u), and the second for the move across (v, u) when that instance of v (there may be several instances) is popped from the stack. □

As noted above, with jumping it would be easy to directly implement depth-first search in $O(m)$ time using $O(n)$ pebbles, and space $O(n \log n)$ in total. The algorithm presented in the proof of Theorem 15 uses more space, namely $\Theta(m \log n)$, since it constructs a representation of the entire graph. It is not known whether the result can be strengthened to match the bounds attained by depth-first search while retaining the weaker model assumed in Theorem 15.

5. Lower Bounds for Active Pebbles

In this section we prove time lower bounds for automata with P active pebbles, but no jumping. The proof generalizes an unpublished construction of Szemerédi (communicated to us by Sipser), that proved an $\Omega(n \log n)$ lower bound on the length of universal traversal sequences for 3-regular graphs.

Theorem 16: Let P and d be fixed functions of n with dn even, $P \geq 1$, $d \geq 6$, and $d^2 + Pd = o(n)$. Let $m = dn/2$, $\epsilon = 1/(3 \ln(6e))$, and

$$d_0 = (2P/e)^{3P/(3P+2)}\, n^{1/(3P+2)}.$$

Let M be any deterministic WAG with P active pebbles that determines st-connectivity for all d-regular n-vertex graphs. Then M requires time:

$$\begin{array}{lll}
(1)\ \ \Omega\left(m(\log n)\frac{d/P}{\log(d/P)}\right), & \text{if } P \le \epsilon \ln(n/d^2) \text{ and } 6P \le d \le d_0, \\
(2)\ \ \Omega\left(mP\left(\frac{n}{d^2}\right)^{\frac{1}{3P}}\right), & \text{if } P \le \epsilon \ln(n/d^2) \text{ and } d_0 < d, \text{ and} \\
(3)\ \ \Omega\left(m \min\left(d, \log \frac{n}{(d^2+Pd)}\right)\right), & \text{otherwise.}
\end{array}$$

Before proving the Theorem, we will make a few observations about it. Perhaps the most noteworthy is that these bounds are nonlinear whenever either $d = \omega(1)$ or $d \ge 6P$, and are better than the simple lower bound given in Theorem 3 except for small constant values of d.

It is obvious that the regions (i.e., the sets of (P, d) pairs) where the three cases apply are pairwise disjoint. It is also true that all three regions are nonempty for all sufficiently large n, although we will not justify this statement.

Although they have very different forms, the three bounds meet "smoothly," except along the line segment $d = 6P$, $1 \le P \le \epsilon \ln(n/d^2)$. Specifically, we will show that where any pair of the three bounds meet along the curve $P = \epsilon \ln(n/d^2)$, $d \ge 6P$, both are $\Theta(m \log(n/d^2))$, and where bounds (1) and (2) meet along the curve $d = d_0$, $1 \le P \le \epsilon \ln(n/d^2)$, both are $\Theta(md_0)$.

All three bounds are increasing functions of d (recall $m = dn/2$). In view of the weak $\Omega(m)$ lower bound given in Theorem 3, the ratio of the lower bounds to m is also an interesting quantity. Note that the ratio of bound (1) to m is an increasing function of d, while that of bound (2) is decreasing. Since they are equal (within constant factors) at $d = d_0$, the two could be combined into the single expression $\Omega(m \min((\log n)(d/P)/\log(d/P), P(n/d^2)^{1/(3P)}))$, as was done in bound (3).

It seems likely that the decrease in bound (2) is an artifact of the proof technique rather than an intrinsic reduction in the complexity of the problem, since intuitively higher degree would seem to make the search more difficult. On the other hand, higher degree reduces the graph's maximum possible diameter, which perhaps helps. It is known that the length of universal traversal sequences is not monotonic in d, although it may be monotonic up to some large threshold, perhaps $d = \lfloor n/2 \rfloor - 1$. (See Borodin *et al.* [14] for a discussion.) Similarly, the complexity of st-connectivity is not monotone in d, since regular graphs of degree $d > \lfloor n/2 \rfloor - 1$ are necessarily connected, but it is plausibly monotone for d up to cn, for some constant $0 < c < 1/2$.

Two special cases of the Theorem are of particular interest. Namely, the following two corollaries show that logarithmic space implies time $m^{1+\Omega(1)}$, and that sublinear space implies superlinear time.

Corollary 17: Let M be a deterministic WAG with P active pebbles that determines st-connectivity for all regular n-vertex graphs. If $P = O(1)$, then there is a family of regular graphs on which M requires time $\Omega(m^{1+1/(3P+3)})$.

Proof: Consider the family of regular graphs with degree $d = d_0 = \Theta(n^{1/(3P+2)})$. Theorem 16 applies, specifically case (1). This gives a time lower bound of $\Omega(md) = \Omega(m^{1+1/(3P+3)})$. □

For $P = 1$ the $\Omega(m^{7/6})$ bound given above is not as strong as the $\Omega(m^2)$ bound given by Borodin *et al.* [14], nor as strong as the $\Omega(n^2)$ bound given in Theorem 11, but is included for comparative purposes. Also, the $\Omega(m^2)$ lower bound for universal traversal sequences holds for degree up to $n/3 - 2$, so the decrease in the ratio of bound (2) to m noted above certainly *is* an artifact of our proof when $P = 1$.

Corollary 18: Let M be a deterministic WAG with P active pebbles that determines st-connectivity for all regular n-vertex graphs. If $P = o(n)$, then there is a family of regular graphs on which M requires time $\Omega(m \log(n/P)) = \omega(m)$.

Proof: Suppose $P \geq n^{1/3}$. Consider the family of regular graphs with degree $d = \sqrt{n/P} = \omega(1)$. Then $d^2 + Pd \leq 2\sqrt{Pn} = o(n)$, so Theorem 16 applies, specifically case (3). This gives a lower bound of $\Omega(m \log d) = \Omega(m \log(n/P))$ on time. When $P < n^{1/3}$, a similar analysis suffices, choosing $d = \log n$. □

Note that by Theorem 15, Corollary 18 is tight: time $O(m)$ is possible with $O(n)$ pebbles. Note also that, when $P = \Theta(n)$, the time is still $\Omega(m \log(n/P))$, by Theorem 3.

Various constants in the Theorem can be improved by slight modification to the construction and/or its analysis, but in the interest of clarity we will not present these refinements.

Proof (of Theorem 16): The idea underlying the proof is to build a graph with many copies of some fixed gadgets, each with many "entry points." Since M does not have enough pebbles to mark all the gadgets it has explored, it must spend time re-exploring each gadget from different entry points, or it risks the possibility that one of them might never be fully explored. The crux of the argument is to choose the right gadgets, and to interconnect them so that we can be sure this happens. We use an "adversary" argument to show this. We begin by giving an overview of the argument, followed by more detailed descriptions of the gadgets and adversary strategy, and finally the analysis.

OVERVIEW. Imagine the adversary "growing" the graph as follows. At a general point in the construction, the graph consists of some gadgets that are fully specified except for the interconnections among their "entry point" vertices. The adversary simulates M on this partial graph until M 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 gadget at the other endpoint of this interconnecting edge f. The adversary will pick it so that M will spend a nonnegligible number of steps τ "exploring" the gadget reached through f. The adversary can achieve this for most of the $\Omega(m)$ interconnecting edges, yielding an $\Omega(m\tau)$ lower bound on time. The parameter τ will vary depending on n, P, and d, giving the three lower bounds quoted in the statement of the theorem.

The interconnecting edge f is chosen as follows. Note that no single labeled gadget γ will suffice to keep p "busy" for τ steps. For example, M's very next move of p, say by label a, might be an exit from γ. On the other hand, if the adversary can learn that M's next move of p will be on label a, it can choose some gadget in which label a moves from an entry point *into* the gadget, rather than exiting from it. Similarly, if it can learn the next τ moves by p (and/or other pebbles following p across f), the adversary can choose a gadget in which this whole sequence of moves avoids exiting from the gadget. A key point is that M can sense only very limited facts about the gadget that p

enters when it crosses f. Suppose p has just crossed f, arriving at a vertex v. M can sense (i) the degree of v, (ii) whether v is the target vertex t, and (iii) whether there are other pebbles on v. Thus, in general M has several possible next moves for p, based on which of these conditions hold. The adversary avoids having to consider these alternative futures by assuming respectively (1) that the graph is d-regular, (2) that M does not reach t (within $\Omega(m\tau)$ steps), and (3) that f connects to a gadget that contains no other pebbles when p enters it, and that remains free of other pebbles (except perhaps ones that follow p across f) for τ moves. Given these assumptions, the adversary will be able to deterministically simulate the next several moves by M so that it can decide which labeled gadget can host those moves without allowing a pebble to exit. Of course, the adversary must also ensure that assumptions (1) – (3) are ultimately justified. Building a d-regular graph requires some care, but is not too difficult. Assumption (2) will follow easily if each connecting edge accounts for τ moves. Assumption (3) is slightly trickier; we will return to it below.

We view the overall adversary strategy as a two-phase process. A *local* phase determines the internal ("local") structure required of a gadget hosting the next several moves of p so that no pebble will exit this gadget until at least τ moves have been charged to it, starting after p's entry. The basic idea is to use a "lazy, greedy" definition — lazy in that the adversary will not define a labeled half edge in the gadget until just before M needs to move a pebble across it, and greedy in that when such a half edge is defined, it will be defined to stay within the gadget. Of course, this cannot continue indefinitely, but will be possible for at least the first τ moves within the gadget. Thus, pebble motion across half edges exiting the gadget is deferred for at least this long.

The adversary's simulation of M is now "rolled back" to the point at which p crossed f. The *global* phase of the adversary's strategy is to choose a gadget already present in the graph and to connect f to it. Recall that our goal is to reuse each gadget many times, so that the total time spent in it asymptotically exceeds its number of edges. (Occasionally, when all entry points of suitable gadgets have been used, a new copy of the needed gadget will be added. This process terminates when the number of vertices in the graph approaches n.) The gadget chosen for f must match the gadget determined by the local phase, must have an unused entry point to which to connect f, and (before f was connected to it) must have remained free of pebbles from the time when p crossed f until τ moves were charged to it. The "pebble-free" condition ensures assumption (3) above. Such a condition is necessary since, if it were violated, M might encounter "unexpected" pebbles in the chosen gadget, i.e., pebbles not encountered during the simulation in the local phase. This could cause M to deviate from the sequence of moves predicted by the local phase, and so possibly allow p or one of the pebbles that followed it across f to exit from the gadget in fewer that τ moves.

A point we slighted above is that the "τ steps" under discussion are not necessarily consecutive, and are not necessarily all made by p or by pebbles that followed p across f. For example, p's moves after crossing f might be interleaved with moves by some other pebble p' after crossing another undefined edge f' and/or many previously defined connecting edges. In general, the adversary keeps track of these many interleaved activities by charging pebble moves to connecting edges, with the "local phase" for an undefined connecting edge f being the interval between its charge reaching 1 (at the first crossing of f by some pebble) and its charge reaching τ.

The final issue to address is that we want to avoid adding a new copy of a gadget until all entry points of most existing copies have been used. Specifically, we will have at most a fixed number ($P(\tau+2)+d$, to be precise) of "open" copies of each gadget at any time. As noted above, many steps may occur between the first and τ^{th} steps charged to f. During this interval, other pebbles

might touch *all* open copies of the gadget needed for f, leaving no *pebble-free* open gadget to which to connect f. Our solution to this problem is found in the adversary's method of charging pebble moves to edges. Moves in f's gadget are always charged to f. In addition, certain moves touching other gadgets are charged to f also. With this scheme, we can bound both the number of moves that occur in f's gadget, and the number of other gadgets that are touched by pebbles during f's local phase. Thus, no gadget is expected to absorb too many moves, and there will be at least one suitable pebble-free open copy of the needed gadget when f accumulates charge τ.

The construction will "waste" (i.e., not fully utilize the connecting edges of) up to $P(\tau+2)+d$ copies of each gadget. The main constraint that limits τ is that it must be small enough that this waste is small, i.e., $P(\tau+2)+d$ times the number of distinct types of labeled gadgets times the number of connecting edges per gadget is small compared to the total number of connecting edges.

We will now present the construction in more detail. We actually define a sequence of graphs $G_{i,j}$, $0 \leq i \leq \mu$, $0 \leq j$, representing successive phases of the construction. Like τ, the parameter μ varies slightly depending on n, p, and d, but will be $\Theta(m)$ in each case. (The maximum value of j is unimportant, but turns out to be about $P\tau$.) Each graph consists of:

- A set of gadgets, each with the same size S and number L of entry vertices, and a fully defined internal structure and labeling. Each vertex that is not an entry vertex has degree d. There is a fixed $d' \geq 1$ such that each entry vertex has d' edges to neighbors in the same gadget, and up to $d-d'$ *connecting edges* joining it to the entry vertices of other gadgets or proto-gadgets (see below). We will show that $d-d' \geq d/2$, and that $L/S > 1/3$, ensuring at termination that the number of connecting edges is $\Theta(m)$.

- A set of labeled *committed* connecting edges joining entry vertices of gadgets. $G_{i,j}$ will have exactly i committed connecting edges.

- A set of up to P partially labeled *uncommitted* connecting edges, each joining an entry vertex u of some gadget to an entry vertex v of a proto-gadget (see below). The uncommitted half edge from u to v is labeled, but the half edge from v to u is unlabeled.

- A set of up to P partially defined *proto-gadgets.* Like a gadget, a proto-gadget has S vertices, including L entry vertices, but unlike the gadgets, the internal structure of a proto-gadget is in general only partially defined — its vertices may have degree less than d, and its half edges may not be labeled. In particular, only one entry vertex v of each proto-gadget will be incident to a connecting edge, say the uncommitted connecting edge $\{u, v\}$, and, as indicated above, the half edge from v to u will be unlabeled. The proto-gadgets are the tools used in the local phases of the adversary's strategy.

In outline, the adversary's strategy is as follows. The initial graph $G_{0,0}$ consists of one arbitrarily chosen gadget. The start vertex s is an arbitrary vertex in this gadget. For any $G_{i,j}$, the *initial configuration* of M on $G_{i,j}$ consists of M in its start state and all P pebbles on $G_{i,j}$'s copy of s. Associate with each connecting edge of the graph $G_{i,j}$ an integer *charge*, initially zero. The adversary will charge each pebble motion to at most one connecting edge, according to a rule to be given later. It will simulate M starting from M's initial configuration on $G_{i,j}$ until one of the following two things happens. (It simulates M as if all vertices in $G_{i,j}$ were of degree d, even though some are of smaller degree.)

- Suppose M attempts to move a pebble from a vertex u across a half edge labeled a, where no such labeled half edge exists. If u is an entry vertex in some gadget, add a new uncommitted half edge from u labeled a to the entry point v of a new proto-gadget. More precisely, we define $G_{i,j+1}$ to be $G_{i,j}$ plus that half edge and proto-gadget. If u is in some proto-gadget, choose some other vertex v in the *same* proto-gadget (according to a rule to be given later) and add a half edge from u to v labeled a. More precisely, we define $G_{i,j+1}$ to be $G_{i,j}$ plus that half edge (plus a few others, as we will see). The choice of v is not arbitrary; one point we must establish is that there will always be a suitable vertex v when needed. The thrust of this step in the adversary strategy is to keep pebbles "trapped" in proto-gadgets for as long as possible. This portion of the adversary's strategy is the "local" strategy introduced above, so-called because of its focus on the structure *within* a gadget.

- Suppose an uncommitted edge f in $G_{i,j}$ accumulates a charge of τ. In this case, we will convert f into a committed edge. More precisely, we will form $G_{i+1,0}$ from $G_{i,j}$ by choosing an existing gadget "similar" to f's proto-gadget, and committing f to enter the chosen gadget. (This is described more fully below.) Again, f cannot be committed arbitrarily; a second point that we must establish is that an appropriate gadget (usually) exists when needed, and that M's behaviors on $G_{i,j}$ and $G_{i+1,0}$ are similar. The thrust of this step is that the size of $G_{i+1,0}$ is growing slowly with i, since we are (usually) able to reuse existing gadgets, but the time M spends in $G_{i+1,0}$ is rising rapidly with i, since a lower bound on the total running time of M is τ times the number of committed edges (i, which is ultimately $\mu = \Theta(m)$). This portion of the adversary's strategy is the "global" strategy, so-called because of its focus on the interconnections among gadgets.

The adversary continues the simulation on $G_{i,j+1}$ or $G_{i+1,0}$ as appropriate, and repeats this process until $G_{\mu,0}$ is constructed.

GADGETS. Before describing the adversary strategy in more detail, we will describe the gadgets and proto-gadgets. The gadgets are called "funnels." An example is shown in Figure 1(a). The entry vertices are those on the "rim" of the funnel. Intuitively, the adversary will try to "trap" pebbles in a funnel for a while by assigning edge labels so that the moves taken by pebbles in the gadget in the near future (i.e., the next τ moves in the gadget) either stay on the same layer or drop to the next deeper layer. The "cone" portion of the funnel (near the top of Figure 1(a)) allows many entry vertices to share vertices in the narrower portion near the bottom of Figure 1(a). An example of a two layer funnel is shown in Figure 1(b).

Four interrelated parameters k, q, g, and r, which in turn depend on n, P, and d, characterize the gadgets. All four are positive integers. Each gadget has $k+1$ *layers*, numbered 0 through k. Layer l, $0 \le l \le k$, has

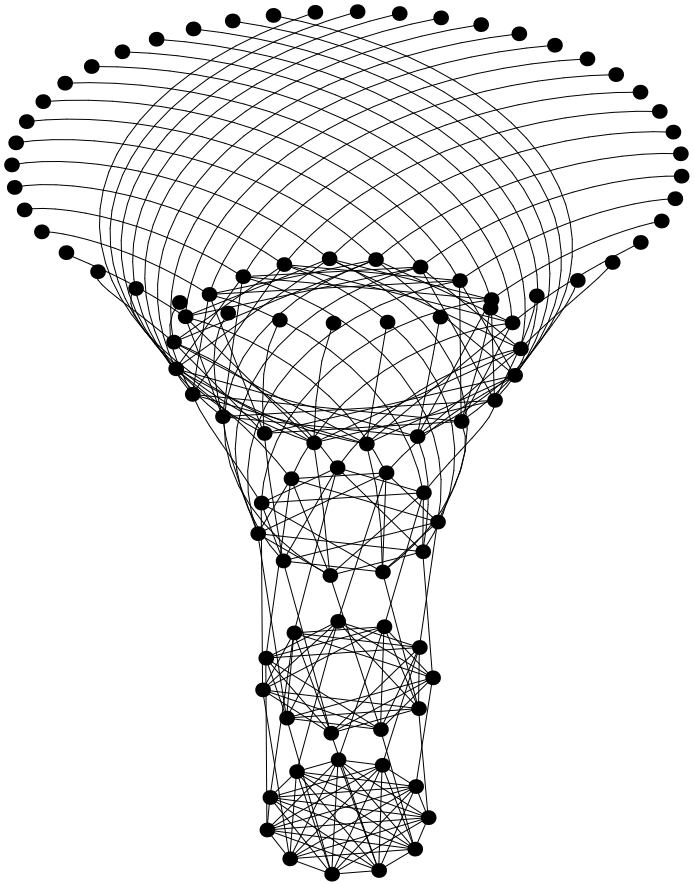
$$n_l = (d+1) \cdot \max(1, 2^{\lceil \log_2 k \rceil - l})$$

vertices, designated v_i^l, $0 \le i \le n_l - 1$. The entry vertices are those on layer 0. Hence, the number of entry vertices is

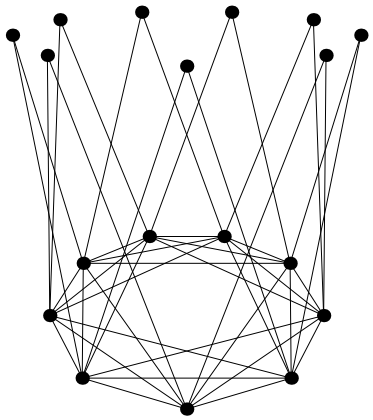
$$L = (d+1) \cdot 2^{\lceil \log_2 k \rceil},$$

and the total number of vertices per gadget is

$$S = (d+1)(2^{\lceil \log_2 k \rceil + 1} - 1 + k - \lceil \log_2 k \rceil).$$



(a)
$d = 10,\ k = 4,\ q = 2,\ r = 1$
For clarity, only half of the forward edges are shown.



(b)
$d = 8,\ k = 1,\ q = 2,\ r = 1$

Figure 1: Examples of the funnel gadgets.

Note that

$$\frac{L}{S} = \frac{(d+1)\cdot 2^{\lceil \log_2 k \rceil}}{(d+1)(2^{\lceil \log_2 k \rceil + 1} - 1 + k - \lceil \log_2 k \rceil)} > \frac{2^{\lceil \log_2 k \rceil}}{2^{\lceil \log_2 k \rceil + 1} + k} \geq \frac{1}{3}, \tag{1}$$

as promised, and that

$$S \leq (1 + 1/d)\, d\, (2 \cdot 2^{\lceil \log_2 k \rceil} + k) \leq (7/6)\, d\, (5k) < 6dk. \tag{2}$$

The parameter q is an even integer, $2 \leq q$. The d edge labels $\{0, 1, \ldots, d-1\}$ are partitioned into $g = \lfloor d/q \rfloor$ "full" blocks, each of size q, plus perhaps one "partial" block of size d mod q in case q does not evenly divide d. The same fixed partition is used for all gadgets and is arbitrary, except that for each $a \in \{0, 1, \ldots, d-1\}$, we place both a and its mate in the same block, where the *mate* of label a is $d - 1 - a$. Note that if d is odd, then $(d-1)/2$ is its own mate, and will be in the partial block.

layer	0	1	$\cdots$	$\lceil \log_2 k \rceil$	$\lceil \log_2 k \rceil + 1$	$\cdots$	$k-1$	k
f_l	r	r	$\cdots$	r	r	$\cdots$	r	0
b_l	$g-r$	$2r$	$\cdots$	$2r$	r	$\cdots$	r	r
cross	0	$g-3r$	$\cdots$	$g-3r$	$g-2r$	$\cdots$	$g-2r$	$g-r$

Table 1: Number of edge blocks of each type per layer.

The remaining gadget parameter r is an integer satisfying $1 \leq r \leq g/3$. Note that the existence of such an r implies that $g \geq 3$, and hence

$$q \leq d/3. \tag{3}$$

Intuitively, r denotes an upper bound on the number of pebbles that we attempt to trap in a given gadget.

The edges within a gadget always connect vertices on the same or adjacent layers. A half edge is called a "forward" half edge if it goes from layer l to layer $l+1$, "backward" if it goes to layer $l-1$, and "cross" if it goes to layer l. For each layer l and each block B of labels, there is a $t \in \{\text{forward}, \text{backward}, \text{cross}\}$ such that all half edges with labels in B leaving vertices on layer l will be of type t. Thus it is natural to refer to the labels and the blocks of labels at a layer as forward, backward, or cross, as well as the half edges. For $i \in \mathbb{N}$, $a \in \{0, 1, \ldots, d-1\}$, and $0 \leq l \leq k$, define

$$\chi(i,a,l) = \begin{cases} (i+a+1) \bmod n_l & \text{if } a < (d-1)/2 \\ (i+n_l/2) \bmod n_l & \text{if } a = (d-1)/2 \\ (i-(d-1-a)-1) \bmod n_l & \text{if } a > (d-1)/2. \end{cases}$$

If $a \in \{0, 1, \ldots, d-1\}$ is a forward label at layer l, then for $0 \leq i \leq n_l - 1$, a will label the half edge from vertex v_i^l to vertex $v_{\chi(i,a,l+1)}^{l+1}$. Similarly, if a is a cross label it will go to $v_{\chi(i,a,l)}^{l}$. Notice that a cross edge labeled a will be labeled by a's mate in the reverse direction. No parallel edges arise since $n_l \geq d+1$. As an example, if all edges are cross edges (a case that does not arise in our constructions) and if $n_l = d+1$, then layer l would be a $(d+1)$-clique. As another example, whenever label 0 is a cross label at layer l, the half edges labeled 0 will form a Hamiltonian cycle through the layer l vertices, and those edges will be labeled $d-1$ (0's mate) in the other direction. Note that the backward labels are not constrained by χ.

The set of gadgets is defined as follows. For $0 \leq l \leq k$ let

$$b_l = \begin{cases} g-r & \text{if } l = 0 \\ r(n_{l-1}/n_l) & \text{otherwise,} \end{cases}$$

$$f_l = \begin{cases} 0 & \text{if } l = k \\ r & \text{otherwise.} \end{cases}$$

See Table 1. If q does not evenly divide d, then the labels in the partial block will be cross labels at each layer $0 \leq l \leq k$. For each layer $0 \leq l \leq k$, choose f_l of the remaining g blocks as forward

labels, and b_l as backward labels (connecting edge labels, if $l = 0$). All blocks not selected above will be cross labels. Note that the rules in the previous paragraph define the forward and cross half edges, given their labels, but not the backward half edges. The chosen backward labels are assigned to these half edges in an arbitrary but fixed way. Note that there are just enough backward labels — each of the n_{l-1} vertices on level $0 \leq l-1 < k$ has exactly qr forward labels, with destinations evenly distributed over the n_l vertices on layer l, so each vertex on layer l is incident to exactly $qr(n_{l-1}/nl) = q \cdot b_l$ edges from layer $l-1$.

For layer 0, the b_0 blocks selected above will label connecting edges. Thus, each entry vertex will be adjacent to exactly $d' = rq + (d \bmod q)$ other vertices in the same gadget, and to $d - d'$ connecting edges. Note, since $r \leq g/3$ and $q \leq d/3$ (from Inequality (3)), that

$$d - d' = gq - rq \geq (2/3)gq = (2/3)\lfloor d/q \rfloor q \geq (2/3)((3/4)(d/q))q = d/2, \tag{4}$$

as claimed earlier. Also note that at most $3r$ blocks are chosen as forward and backward at each layer, and that this is always possible since $g \geq 3r$.

The number of distinct gadget types created by this process is

$$\begin{aligned}
&\binom{g}{r}^k \binom{g-r}{2r}^{\lceil \log_2 k \rceil} \binom{g-r}{r}^{k-\lceil \log_2 k \rceil - 1} \binom{g}{r}^1 \\
&\qquad \leq \binom{g}{r}^{2k-\lceil \log_2 k \rceil} \binom{g}{2r}^{\lceil \log_2 k \rceil} \\
&\qquad \leq \left(\frac{eg}{r}\right)^{r(2k-\lceil \log_2 k \rceil)} \left(\frac{eg}{2r}\right)^{2r\lceil \log_2 k \rceil} \\
&\qquad \leq \left(\frac{eg}{r}\right)^{r(2k+\lceil \log_2 k \rceil)}.
\end{aligned} \tag{5}$$

Figure 1(b) fully shows a gadget with $d = 8$, $k = 1$, $q = 2$, $g = 4$, and $r = 1$, with forward edges labeled 0 and 7 from layer 0, and backward edges labeled 3 and 4 from layer 1. Figure 1(a) shows a gadget with $d = 10$, $k = 4$, $q = 2$, $g = 5$, and $r = 1$, with forward edges labeled 0 from layers 0 through 3. In the interest of clarity, the forward edges labeled 9 (0's mate) are not shown in the figure.

Proto-Gadgets and Local Strategy. The proto-gadgets are built incrementally by the adversary. Initially, each consists of S vertices, denoted as in the gadgets, together with the cross edges defined by the partial block of labels (if any) at each level. As discussed previously, the adversary proceeds by simulating M from its initial configuration on $G_{i,j}$. Suppose during the t^{th} step of this simulation that M attempts to move some pebble p along the half edge labeled a from some vertex u but no such half edge exists. As sketched earlier, if u is an entry vertex of some gadget, we create a new proto-gadget into which p will move. If u is a vertex v_i^l in some proto-gadget π, the adversary decides whether to make the block of labels containing a all forward half edges or all cross half edges (see below). The graph $G_{i,j+1}$ is then defined to be the same as $G_{i,j}$, except that at layer l in π, a's block of half edges are added. The adversary restarts the simulation of M, starting from M's initial configuration on $G_{i,j+1}$. It should be clear that during the first $t-1$ steps of the simulation, M will behave on $G_{i,j+1}$ exactly as it did on $G_{i,j}$, since $G_{i,j}$

is a subgraph of $G_{i,j+1}$. The t^{th} step, of course, was impossible in $G_{i,j}$, but is possible in $G_{i,j+1}$. Note that p can exit from π only at an entry vertex, but is no nearer to one in $G_{i,j+1}$ after the t^{th} step than before. Thus we can view M as running on a dynamically growing graph, one being built by the adversary so as to trap pebbles in proto-gadgets for some number of moves. We will adopt this view when no confusion will arise, and let $G_{i,*}$ denote the last $G_{i,j}$ built before $G_{i+1,0}$.

Let z be the number of free blocks at level l, i.e., blocks whose half edges have not yet been defined. The adversary chooses a's block to label cross edges provided $z > b_l + f_l$, and forward edges provided $b_l < z \leq b_l + f_l$. If $z \leq b_l$, the adversary *fails* (but see Claim 1 below).

Let

$$\tau = \begin{cases} (k - \lceil \log_2 k \rceil)\, \lfloor g/3 \rfloor & \text{if } P \leq r \\ r & \text{if } P > r. \end{cases}$$

Note that $(k - \lceil \log_2 k \rceil)\, \lfloor g/3 \rfloor \geq r$ since $k \geq 1$ and $g/3 \geq r$, so in either case we have

$$(k - \lceil \log_2 k \rceil)\, \lfloor g/3 \rfloor \geq \tau. \tag{6}$$

We prove three claims about the proto-gadgets. We will see later that the global strategy prevents M from making more than τ moves in any proto-gadget, so Claim 1 below shows that the adversary will never fail.

Claim 1: The adversary will never fail, provided M makes at most τ moves in any proto-gadget.

Proof: First, clearly at most $\min(P, \tau)$ pebbles can enter a proto-gadget in τ steps, and for the particular definition of τ chosen above, $\min(P, \tau) \leq r$. Now, suppose the claim is false. Suppose the adversary first fails during an attempted move at level l in some proto-gadget π. Then at least $g - b_l$ moves were previously made by pebbles at layer l. As noted, at most r pebbles can enter π in τ moves. It cannot be the case that $l < k$, since for all such layers $g - b_l \geq r = f_l$, so during the last r of the $g - b_l$ moves, all r pebbles moved past layer l, leaving none to cause failure there. Thus, the failure occurred in layer k. For a pebble to reach layer k, it must be that the maximum number of cross edges, plus at least one forward edge, have been previously defined at each layer less than k. Thus, the number of moves completed in this proto-gadget prior to failure is at least

$$\begin{aligned} & (g - b_k) + \sum_{l=0}^{k-1} (g - b_l - f_l + 1) \\ &= (g - r) + \lceil \log_2 k \rceil\, (g - 3r + 1) + (k - \lceil \log_2 k \rceil - 1)(g - 2r + 1) \\ &\geq (k - \lceil \log_2 k \rceil)(g - 2r + 1) \\ &> (k - \lceil \log_2 k \rceil)\, \lfloor g/3 \rfloor \\ &\geq \tau. \end{aligned}$$

The second inequality uses the assertion that $r \leq g/3$, and the third uses Inequality (6). □

Claim 2: Each proto-gadget is a subgraph of some gadget.

Proof: The adversary chooses at most f_l forward blocks and at most $g - (f_l + b_l)$ cross blocks at each layer. Thus there are enough unchosen blocks to select a total of exactly f_l forward and b_l backward blocks, which precisely defines a gadget. □

Claim 3: All entry points of a proto-gadget are equivalent, in the sense that if M makes at most τ moves in a proto-gadget entered through vertex v_h^0, then the resulting configuration will be exactly the same as if it had entered through vertex v_0^0, except that positions of all pebbles in it on layer l will be shifted by $h \bmod n_l$ for $0 \leq l \leq k$.

Proof: Intuitively, this reflects the rotational symmetry of the funnel. To make this precise, we claim that for any $h \in \mathbb{N}$ and any proto-gadget π, the mapping $\phi_h(v_i^l) = v_{i'}^l$, where $i' = (i+h) \bmod n_l$, is an automorphism on π, i.e., a surjection on the vertices of π preserving labeled half-edges. Consider a forward edge labeled a at level l in π, say (v_i^l, v_j^{l+1}), where $j = \chi(i, a, l+1)$. Note that for each fixed a and l, there is a constant c (depending on a and l but independent of i) such that $\chi(i, a, l+1) = (i + c) \bmod n_{l+1}$. Now $\phi_h(v_i^l) = v_{i'}^l$, $\phi_h(v_j^{l+1}) = v_{j'}^{l+1}$, with $i' = (i + h) \bmod n_l$, and $j' = (j + h) \bmod n_{l+1}$, so since n_{l+1} divides n_l we have

$$\begin{aligned}
\chi(i', a, l+1) &= (i' + c) \bmod n_{l+1} \\
&= (((i + h) \bmod n_l) + c) \bmod n_{l+1} \\
&= (i + h + c) \bmod n_{l+1} \\
&= (((i + c) \bmod n_{l+1}) + h) \bmod n_{l+1} \\
&= (j + h) \bmod n_{l+1} \\
&= j'.
\end{aligned}$$

A similar argument applies to cross edges. □

The analog of Claim 3 also holds for gadgets, provided the τ moves use only forward and/or cross edges. The same may not be true if backward edge labels are used.

GLOBAL STRATEGY. We have now described the gadgets and proto-gadgets, and the adversary's strategy for building them. We turn to the remaining part of its strategy — charging and committing edges. Recall that the adversary associates a charge with each connecting edge, in which it counts moves in $G_{i,*}$. In addition, it associates with each connecting edge a second integer, called a *birthdate*, recording the time at which a pebble first crosses the edge.

The construction of $G_{i+1,0}$ from $G_{i,*}$ proceeds as follows. The adversary begins with M in its *initial* configuration in the current graph $G_{i,*}$. The adversary simulates 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 proto-gadget. Let f be the connecting edge most recently crossed by p. If f has charge less than τ, then charge the move to f; otherwise there is no charge.

- a connecting edge f (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 f in either direction, define the birthdate of f to be the current time.

As sketched in the overview, the second charging rule ensures that when an uncommitted connecting edge f, even one whose associated pebbles have moved infrequently, accumulates charge τ, only a few of the gadgets of the appropriate type can have been touched by pebbles since the birth of f.

When some uncommitted connecting edge $f = \{u, v\}$ with label $\lambda_{u,v} = a$ accumulates charge τ we stop the simulation, and construct from $G_{i,*}$ a new graph $G_{i+1,0}$ defined as follows. Let π_v be the proto-gadget entered through f, with v in π_v. Note that by the charging rules above, each move in π_v has been charged to f, so there have been at most τ such moves. Thus by Claim 1 the adversary did not fail while building π_v. By Claim 2, the proto-gadget π_v is a subgraph of some gadget γ_v. We say an entry vertex of a gadget is *open* if it has degree less than d. If possible, choose an entry vertex x of a gadget in $G_{i,*}$ such that

$$\left.\begin{array}{l} \bullet\ x \text{ is open,} \\ \bullet\ x\text{'s gadget is of the same type as } \gamma_v, \\ \bullet\ x \text{ and } u \text{ are not adjacent, and} \\ \bullet\ x\text{'s gadget has remained pebble free since the birthdate of the uncommitted edge } f. \end{array}\right\} \quad (7)$$

$G_{i+1,0}$ is identical to $G_{i,*}$, except that the proto-gadget π_v is removed, and the uncommitted edge $f = \{u, v\}$ is replaced by the committed edge $\{u, x\}$ with labels $\lambda_{u,x} = a$ and $\lambda_{x,u} = b$, where b is any label not already present on an outgoing half edge at x. If there is no such x, or if using the only such x would result in $G_{i+1,0}$ having neither uncommitted edges nor open entry vertices, we instead add one additional gadget of type γ_v, choose as x any of the new gadget's entry vertices, then proceed as described above. The latter contingency avoids premature termination of the construction. The requirement that x and u be nonadjacent avoids construction of parallel edges.

The behavior of M on $G_{i+1,0}$ is similar to its behavior on $G_{i,*}$. Suppose in $G_{i,*}$ the uncommitted edge f was first crossed during the simulation of the b^{th} move of M (i.e., has birthdate b), and accumulates charge τ during move b'. When M is simulated on $G_{i+1,0}$, it will behave exactly as on $G_{i,j}$ for the first $b-1$ moves, since the portion of $G_{i+1,0}$ visited during that period is exactly the same as the portion visited in $G_{i,*}$. In particular, the charges and birthdates attached to edges will be the same. (Thus, one can view the adversary as rolling back the simulation to step b, committing f, and resuming.) Between steps b and b' those pebbles that crossed edge f in $G_{i,*}$ will be in x's gadget γ_x in $G_{i+1,0}$ instead of in the proto-gadget π_v entered through f as they were in $G_{i,*}$, but since γ_x contains π_v as a subgraph, their motions in $G_{i+1,0}$ will exactly reflect their motions in $G_{i,*}$. Note that by Claim 3, this is true regardless of which entry vertex x of γ_x was chosen. It is crucial that the chosen gadget γ_x was pebble free between steps b and b', so there is no possibility that these pebbles will meet pebbles in γ_x in $G_{i+1,0}$ that they did not meet in π_v in $G_{i,*}$. Again, the charges and birthdates attached to edges will be the same in $G_{i+1,0}$ as in $G_{i,*}$ through step b'. In particular, each of the $i+1$ committed edges in $G_{i+1,0}$ will have a charge of τ, and hence M will run for at least $(i+1)\tau$ steps on $G_{i+1,0}$.

Final Construction. After $G_{i+1,0}$ is built, we restart the simulation from the beginning on $G_{i+1,0}$ to build $G_{i+2,0}$, etc. Continue this process until $G_{\mu,0}$ is constructed. Finally, from $G_{\mu,0}$ we

build a pair of similar graphs G and G', one connected and the other not, on which M will have identical behavior. In particular, if M runs for fewer than $\mu \cdot \tau$ steps, then M cannot be correct on both. The connected graph G is built by

- committing all uncommitted edges, as described above,
- joining the remaining open entry vertices with some number, Δ, of extra vertices so as to make G have n vertices and be d-regular, and
- designating one of these extra vertices as t.

One way to accomplish the second step is the following. First, pick any two nonadjacent open vertices, and connect them. Repeat this as often as possible. Let u be the number of "missing" half edges, i.e., the total over all open vertices of d minus their degrees, and let i be the number of remaining open vertices. Since the pairing process could not be applied to reduce i further, it must be the case that the i open entry vertices form a clique. Recalling that each entry vertex is incident to at most $d - d'$ connecting edges, the number u of missing half edges can be at most

$$i((d - d') - (i - 1)) \leq i(d - i) \leq d^2/4,$$

since $d' \geq 1$, and since $i(d - i)$ is maximized when $i = d/2$. Thus $u \leq d^2/4$. Furthermore, u will necessarily be even, since each entry vertex starts with $d - d'$ missing half edges; since from Equation (4) $d - d'$ is a multiple of q, hence even; and since each committed edge replaces a pair of missing half edges. Notice that this implies that $d(n - \Delta)$ is even, since the gadgets together contain $n - \Delta$ vertices and $d(n - \Delta) - u$ half edges, which naturally occur in pairs. Complete the construction by adding a Δ-vertex, d-regular graph that contains a $u/2$-matching, removing the edges of this matching, and connecting each of the u missing half edges to a distinct endpoint of the matching. Such a regular graph exists by Proposition 2, since dn, $d(n - \Delta)$, and hence $d\Delta$ are even; since, as shown below, $d < \Delta$ and $u \leq d^2/4 < \Delta$; and since the proof of Proposition 2 given in Borodin *et al.* [14] constructs a regular graph that is Hamiltonian and hence has a $u/2$-matching. (That construction is similar to the construction of cross edges in one layer of our gadgets, where the 0-labels form a Hamiltonian cycle.)

The nonconnected graph G' is built similarly, except that $d+1$ of the Δ extra vertices, including t, are connected in a clique, and hence disconnected from the rest of the graph.

By an argument similar to one above, M's behavior on both G and G' is essentially the same as on $G_{\mu,0}$. In particular, the edge charges will be the same, so it will run for at least $\mu \cdot \tau$ steps without reaching any of the Δ extra vertices, including t. One point to be shown in the analysis below is that $\Delta \geq d + 1 + \max(d + 1, d^2/4) = d^2/4 + d + 1$, i.e., large enough to allow completion of the construction of G and G' as described above. Since $d \leq \sqrt{n} - 2$ (in fact, $d^2 + Pd = o(n)$), it suffices that $\Delta \geq n/4$.

Analysis. All that remains to show our $\Omega(m\tau)$ lower bound is to give values for the various parameters so as to satisfy the constraints listed above (and to maximize τ). For convenience, we summarize the relevant parameters and constraints here.

C1. Number of committed connecting edges: $\mu = \Omega(m)$.

C2. Number of vertices added in the final step of the construction: $\Delta \geq n/4$.

C3. Number of layers per gadget: $k \geq 1$.

C4. Size of full blocks in the label partition: $q \geq 2$, even.

C5. Number of full label blocks: $g = \lfloor d/q \rfloor$.

C6. Upper bound on the number of pebbles entering a proto-gadget: $1 \leq r \leq g/3$.

C7. Time per committed edge: τ; if $P \leq r$ then $\tau = (k - \lceil \log_2 k \rceil)\, \lfloor g/3 \rfloor$ else $\tau = r$.

To satisfy constraint C1, choose

$$\mu = \lfloor dLn/(8S) \rfloor . \tag{8}$$

Since we have seen in Inequality (1) that $L/S = \Omega(1)$, we have $\mu = \Omega(m)$ as claimed above.

We now turn to constraint C2. We say a gadget is *closed* if each of its L entry vertices is connected to the maximum number $d - d'$ of committed half edges; otherwise the gadget is *open*. $G_{\mu,0}$ has exactly μ committed edges, or 2μ committed half edges. From Inequality (4), each closed gadget contributes $(d-d')L \geq dL/2$ committed half edges, so by Equation (8) there can be no more than $2\mu/(dL/2) \leq n/(2S)$ closed gadgets in $G_{\mu,0}$, each of size S, and so closed gadgets contribute no more than $n/2$ vertices to G. Thus, to ensure constraint C2, i.e., that Δ is at least $n/4$, it suffices to ensure that the following additional constraint holds:

C8. Number of vertices in open gadgets: must be at most $n/4$.

When building $G_{i+1,0}$ from $G_{i,*}$, the adversary replaces a proto-gadget π by a copy of a fixed gadget γ. There might be many copies of the gadget γ with which π can be replaced. A key claim in establishing constraint C8 is that there are never more than $P(\tau+2)+d$ open copies of such a gadget.

Claim 4: When $G_{i+1,0}$ is defined, if there are $P(\tau+2)+d$ open copies of the gadget γ_v, then at least one of them will have an entry vertex x satisfying the conditions (7), so a new (open) gadget will *not* be introduced into $G_{i+1,0}$.

Proof: 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 $d - d' \leq d - 1$ entry vertices are adjacent to u in $G_{i,*}$. A more subtle problem is to bound the number of gadgets that can be touched by pebbles between the birth of π's uncommitted connecting edge f, and the time at which f has accumulated charge τ. At most P gadgets contain pebbles at the time of f's birth. At most $P-1$ edges older than f can have charge less than τ, because, by Claim 1, for each such edge f' there is at least one pebble that does not leave its gadget or proto-gadget until f' has accumulated charge τ. Each gadget touched by some pebble after the birth of f necessitates the crossing of some connecting edge. Thus after at most $(P-1)\tau$ such crossings, f will be the oldest uncommitted edge, and after at most τ more crossings, f will have charge τ. Thus, at most $P(\tau+1)$ gadgets can be touched by pebbles during the relevant interval. Finally, at all times at most P open gadgets are incident to uncommitted half edges, hence at most P lack open entry vertices. Thus, the number of vertices x not disqualified is at least $P(\tau+2)+d-(d-1)-P(\tau+1)-P = 1$, which establishes the claim. □

Inequality (5) bounds the number of distinct gadget types, Claim 4 bounds the number of open copies of each, and Inequality (2) bounds the size of each copy. Thus, the total number of vertices in open gadgets is at most

$$\left(\frac{eg}{r}\right)^{r(2k+\lceil \log_2 k\rceil)} (P(\tau+2)+d)\,6dk. \tag{9}$$

We divide the remainder of the analysis into two cases. The second applies when P is small. The first applies to either small or large P, but gives a weaker bound than the second for small P.

Case 1: Let $\delta = 3\epsilon/2 = 1/(2\ln(6e))$, let

$$\beta = 72\left(d^2 + Pd\ln\frac{n}{d^2+Pd}\right),$$

and suppose n, P, and d are such that $d^2 + Pd \le n/e$ and $\beta \le n/e^{6/\delta}$, both of which are true for all sufficiently large n, since $d^2 + Pd = o(n)$. Then we claim that the following parameter values satisfy constraints C3–C8.

$$\begin{aligned} k &= 1 \\ q &= 2\left\lceil \frac{d-5}{\delta\ln(n/\beta)}\right\rceil \\ g &= \lfloor d/q\rfloor \\ r &= \lfloor g/3\rfloor \\ \tau &= r \end{aligned}$$

Note that constraints C3 and C5 are immediately satisfied, as is constraint C7 since $k = 1$. It is also immediate that q is even, and is positive, since $d \ge 6$, $\delta > 0$, and $n/\beta > 1$, hence constraint C4 is satisfied.

For constraint C6, it is immediate that $r \le g/3$. To show $r \ge 1$ it suffices to show $q \le d/3$:

$$q = 2\left\lceil \frac{d-5}{\delta\ln(n/\beta)}\right\rceil \le 2\left\lceil \frac{d-5}{6}\right\rceil = 2\left\lfloor \frac{d}{6}\right\rfloor \le \frac{d}{3}.$$

To satisfy constraint C8 above, we first note (making frequent use of the inequalities $x/2 \le \lfloor x\rfloor$ and $\lceil x\rceil \le 2x$, valid for all $x \ge 1$) that

$$\begin{aligned} g/r &= g/\lfloor g/3\rfloor \le g/(g/6) = 6, \\ r &= \lfloor g/3\rfloor \le g/3 = \lfloor d/q\rfloor/3 \le d/(3q) \\ &= \frac{d}{6\left\lceil \frac{d-5}{\delta\ln(n/\beta)}\right\rceil} \le \frac{1}{6}\frac{d}{d-5}\delta\ln(n/\beta) \le \delta\ln(n/\beta), \\ r+2 &\le 3r, \\ d^2+Pd &\le d^2 + Pd\ln\frac{n}{d^2+Pd} < \beta, \text{ and} \\ \delta &= 1/(2\ln(6e)) < 1. \end{aligned}$$

Returning to constraint C8, we must show that Expression (9) is at most $n/4$:

$$\begin{aligned}
&\left(\frac{eg}{r}\right)^{r(2k+\lceil \log_2 k \rceil)} (P(\tau+2)+d)\,6dk \\
&\qquad = 6\left(\frac{eg}{r}\right)^{2r}(d^2+Pd(r+2)) \\
&\qquad \leq 18(6e)^{2r}(d^2+Pdr) \\
&\qquad \leq 18(6e)^{2\delta\ln(n/\beta)}(d^2+\delta Pd\ln(n/\beta)) \\
&\qquad = 18(n/\beta)(d^2+\delta Pd\ln(n/\beta)) \\
&\qquad = \frac{18n(d^2+\delta Pd\ln(n/\beta)}{72\left(d^2+Pd\ln\frac{n}{d^2+Pd}\right)} \\
&\qquad < n/4,
\end{aligned}$$

as desired.

To complete the analysis of case 1, we show that τ is large enough to imply the bound in the statement of the theorem:

$$\begin{aligned}
\tau &= r = \lfloor g/3 \rfloor \geq g/6 = \lfloor d/q \rfloor /6 \geq d/(12q) \\
&= \frac{d}{24\left\lceil \frac{d-5}{\delta\ln(n/\beta)}\right\rceil}.
\end{aligned}$$

The latter quantity equals $d/24$, if $d \leq \delta\ln(n/\beta)+5$. If $d > \delta\ln(n/\beta)+5$, then:

$$\begin{aligned}
\frac{d}{24\left\lceil \frac{d-5}{\delta\ln(n/\beta)}\right\rceil} &\geq \frac{d}{48\left(\frac{d-5}{\delta\ln(n/\beta)}\right)} \\
&= \frac{1}{48}\frac{d}{d-5}\delta\ln(n/\beta) \\
&\geq \frac{\delta\ln(n/\beta)}{48} \\
&= \frac{\delta}{48}\ln\frac{n}{72\left(d^2+Pd\ln\frac{n}{d^2+Pd}\right)} \\
&\geq \frac{\delta}{48}\ln\frac{n}{72\,(d^2+Pd)\ln\frac{n}{d^2+Pd}} \\
&= \Omega\left(\ln\frac{n}{d^2+Pd}\right).
\end{aligned}$$

The penultimate inequality holds since by assumption $\ln(n/(d^2+Pd)) \geq 1$. The final lower bound follows since $\ln(x/(72\ln x)) = \Omega(\ln x)$. Thus, $\tau = \Omega(\min(d, \ln(n/(d^2+Pd))))$ as claimed in the statement of the theorem.

Case 2: Recall $\epsilon = 1/(3\ln(6e))$, and suppose $6P \le d \le \sqrt{n}/6^9$ and $1 \le P \le \epsilon \ln(n/d^2)$. (Note that $d \le \sqrt{n}/6^9$ must be true for all sufficiently large n, since $d^2 + Pd = o(n)$.) Then we claim that the following parameter values satisfy constraints C3–C8.

$$\begin{aligned}
\hat{q} &= \frac{ed}{P}\left(\frac{d^2}{n}\right)^{1/(3P)} \\
q &= 2\lceil \hat{q}/2 \rceil \\
g &= \lfloor d/q \rfloor \\
r &= P \\
k &= \left\lfloor \frac{\ln(n/d^2)}{3P\ln(ed/(qP))} \right\rfloor \\
\tau &= (k - \lceil \log_2 k \rceil)\lfloor g/3 \rfloor = \Theta(gk)
\end{aligned}$$

Note that constraint C5 is immediately satisfied, as is constraint C7 since $r \ge P$. Since $\hat{q}$ is positive, it is also immediate that q is even and is positive, hence constraint C4 is satisfied.

Note for future use that

$$(n/d^2)^{1/(3P)} \ge (n/d^2)^{1/(3\epsilon\ln(n/d^2))} = e^{1/(3\epsilon)} = 6e. \tag{10}$$

For constraint C3, note that $q \ge \hat{q}$. Thus,

$$k = \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln(ed/(qP))} \right\rfloor \ge \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln(ed/(\hat{q}P))} \right\rfloor = \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln((n/d^2)^{1/(3P)})} \right\rfloor = 1.$$

Thus, $k \ge 1$. Using a similar analysis, we note for future use that $k = 1$ whenever $q > 2$. This holds since $q > 2$ implies $\hat{q}/2 > 1$, which implies $q \le 2\hat{q}$. Thus,

$$k = \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln(ed/(qP))} \right\rfloor \le \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln(ed/(2\hat{q}P))} \right\rfloor = \left\lfloor \frac{\ln((n/d^2)^{1/(3P)})}{\ln((n/d^2)^{1/(3P)}/2)} \right\rfloor = 1. \tag{11}$$

The last equality follows from the fact that $1 < (\ln x)/(\ln(x/2)) < 2$ whenever $x > 4$, and from Inequality (10).

For constraint C6, it is immediate that $r = P \ge 1$. To show $r \le g/3$ it suffices to show $3qP \le d$. If $q = 2$, this holds since by assumption $6P \le d$. If $q > 2$, then $\hat{q}/2 > 1$, so

$$3qP = 6\lceil \hat{q}/2 \rceil P \le 6\hat{q}P = 6(ed/P)(d^2/n)^{1/(3P)}P \le 6ed/(6e) = d. \tag{12}$$

The last inequality follows from Inequality (10).

For constraint C8, we first note for integer $k \ge 1$ that

$$2k + \lceil \log_2 k \rceil \le 8k/3.$$

(The bound is tight at $k = 3$.) Also,

$$\tau + 2 = (k - \lceil \log_2 k \rceil)\lfloor g/3 \rfloor + 2 \le gk,$$

since $g \geq 3$. Using Expression (9), we bound the number of vertices in open gadgets as follows.

$$\begin{aligned}
&\left(\frac{eg}{r}\right)^{r(2k+\lceil \log_2 k \rceil)} (P(\tau+2)+d)\,6dk \\
&\quad \leq \; 6\left(\frac{eg}{P}\right)^{8Pk/3} dk(Pgk+d) \\
&\quad \leq \; 6\left(\frac{ed}{qP}\right)^{(8/3)P\left\lfloor \frac{\ln(n/d^2)}{3P\ln(ed/(qP))}\right\rfloor} dk(Pdk/q+d) \\
&\quad \leq \; 6\left(\frac{n}{d^2}\right)^{8/9} d^2(Pk^2/q+k) \\
&\quad \leq \; 6n^{8/9}d^{2/9}(Pk^2/q+k)
\end{aligned}$$

We break the rest of the derivation of constraint C8 into two subcases based on d. Note that, since $3qP \leq d$ from Inequality (12),

$$k = \left\lfloor \frac{\ln(n/d^2)}{3P\ln(ed/(qP))} \right\rfloor \leq \frac{\ln n}{3P\ln(3e)} \leq \ln n.$$

When $d < n^{1/4}$, since k and P are both $O(\log n)$, we have

$$6n^{8/9}d^{2/9}(Pk^2/q+k) = O(n^{8/9}(n^{1/4})^{2/9}\log^3 n) = O(n^{17/18}\log^3 n) = o(n).$$

When $d \geq n^{1/4}$, we will show that $q > 2P$ and $k = 1$, so we have

$$6n^{8/9}d^{2/9}(Pk^2/q+k) \leq 6n^{8/9}(n^{1/2}/6^9)^{2/9}(1/2+1) = n/4.$$

We show that $q > 2P$ as follows.

$$\frac{q}{P} \geq \frac{\hat{q}}{P} = \frac{ed}{P^2}\left(\frac{d^2}{n}\right)^{1/(3P)} \geq \frac{n^{1/4}}{P^2}\left(\frac{n^{2/4}}{n}\right)^{1/3} = \frac{n^{1/12}}{P^2} = \omega(1).$$

(Recall $P = O(\log n)$.) Since $q > 2P$ and $P \geq 1$, we have $q > 2$, so $k = 1$ by Equation (11).

To complete the analysis of Case 2, we show that $\tau = (k - \lceil \log_2 k \rceil)\lfloor g/3 \rfloor$ is large enough to imply the bound in the statement of the theorem. Again we split the analysis into two subcases based on d. We have $q > 2$ if and only if $\hat{q} > 2$, which holds exactly when

$$d > d_0 = (2P/e)^{3P/(3P+2)}\, n^{1/(3P+2)}.$$

In this case we have $k = 1$ by Equation (11), and

$$\begin{aligned}
\tau &= \lfloor g/3 \rfloor \geq g/6 \geq d/(12q) \geq d/(24\hat{q}) = \frac{d}{24\left(\frac{ed}{P}\left(\frac{d^2}{n}\right)^{1/(3P)}\right)} \\
&= \frac{P}{24e}\left(\frac{n}{d^2}\right)^{1/(3P)} \qquad (13) \\
&= \Omega\left(P\left(\frac{n}{d^2}\right)^{1/(3P)}\right),
\end{aligned}$$

as claimed. We remark that, by Equation (10), when P is maximal, Expression (13) is $P/4 = \Theta(\log(n/d^2))$, so the transition to the bound given in Case 1 is "smooth."

In the second subcase we have $d \leq d_0$. First, note that

$$d_0 = (2P/e)^{3P/(3P+2)}\, n^{1/(3P+2)} \leq P n^{1/(3P+2)} \leq P n^{1/5} = o(n^{1/4}). \tag{14}$$

Second, since $d \leq d_0$, we have $q = 2$ and $k \geq 1$. Also, note that $(k - \lceil \log_2 k \rceil)/k \geq 1/3$, (attaining the minimum at $k = 3$) and that $g \geq 3$. Hence $\tau = \Omega(gk)$, and

$$\begin{aligned} gk &= \left\lfloor \frac{d}{q} \right\rfloor \left\lfloor \frac{\ln(n/d^2)}{3P \ln(ed/(qP))} \right\rfloor \\ &\geq \frac{d \ln(n/d^2)}{24P \ln(ed/(2P))} \\ &= \frac{d \ln(n/o(n^{1/2}))}{24P \ln(ed/(2P))} \\ &= \Omega\left(\frac{d/P}{\ln(d/P)} \ln n \right), \end{aligned}$$

as claimed. We remark that when P is maximal and $d \geq 6P$, the estimate in Inequality (14) can be refined, allowing one to show $d = \Theta(P) = \Theta(\log n)$. Thus, τ again matches the bound in Case 1 (up to constant factors).

Finally, when $d = d_0$ we have $\hat{q}$ exactly equal to 2; similarly, when $d = d_0$ the expression of which k is the floor is precisely 1. Furthermore, both expressions vary slowly with d, so both are $\Theta(1)$ when d is near d_0. Thus, again $\tau = (k - \lceil \log_2 k \rceil)\, \lfloor \lfloor d/q \rfloor /3) \rfloor$ is "smooth" as d crosses d_0, the threshold between the lower bounds quoted in (1) and (2) in the statement of the Theorem, and in fact both lower bounds are $\Theta(md_0)$ for d near d_0.

This completes the proof. □

It is interesting to note why the proof would fail if M were allowed to jump pebbles. In the local phase, the adversary was 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 $\Theta(d)$, this has touched all $\Theta(d)$ connecting edges incident to that entry vertex, which was impossible in the construction above.

The results given in this section are a first step towards understanding the power of JAGs with active pebbles. The results focus on graph families with nonbijective labeling and are strongest when degree is larger than the number of pebbles, and is growing. Especially in light of Lemma 1, it would be desirable to extend the results to the interesting case of symmetrically labeled graphs of fixed degree, say $d = 3$, with a number of pebbles greater than the degree.

6. Lower Bounds for the Cycle

6.1. A Lower Bound on the Number of States

In this section we show that deterministic 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.

Lemma 19: 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 v on C_α that is an even distance from L and traversing according to α terminates at M.

Proof: Consider three pebbles traversing simultaneously according to α, beginning at L, v, and R, respectively. A straightforward induction shows that the pebble that began at v is always an even distance from the other two and between them. Since the ones that started at L and R both end at M, so does the third. □

Theorem 20: Any WAG W that traverses every labeled n-cycle using Q states, one active pebble, and P passive pebbles satisfies $(P+4)Q \geq n$.

Proof: Assume to the contrary that $(P+4)Q < n$. Consider the action of W's active pebble if it never encounters a passive pebble it previously dropped: it traverses according to the sequence $t = \alpha_0\alpha_1\cdots\alpha_P \in \{0,1\}^*$, where α_i is its traversal after dropping i but before dropping $i+1$ pebbles. If each $|\alpha_i| \leq Q$, then $|t| \leq (P+1)Q < n-1$, so that W does not traverse any cycle having t as the prefix of the clockwise labeling beginning at the start vertex. Thus let i be the least integer such that $|\alpha_i| > Q$. Then W repeats some state during this interval, and $\alpha_i = \rho\beta\beta\beta\cdots$ is infinite, with $|\rho| + |\beta| \leq Q$.

Let $\alpha = \beta\beta$ and $t' = \alpha_0\alpha_1\cdots\alpha_{i-1}\rho$. Consider the cycle in which t' is the clockwise labeling from the start vertex to a vertex L, followed by an embedding of the chain C_α of Lemma 19 from L clockwise to R. Notice that $|t'| + |C_\alpha| \leq (P+4)Q < n$, so that this labeling can be embedded on a cycle of length n. Now a traversal according to $t'\alpha$ causes the active pebble to move unidirectionally to the midpoint M of C_α, so that no pebble dropped is reencountered. By Lemma 19, each further traversal according to α returns to M, so that the pebbles previously dropped cannot be reencountered, and R is never reached. □

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. The idea is to maintain the invariant that the leading and trailing pebbles are on adjacent vertices, and the automaton knows the label from the trailing pebble to the leading pebble. Now after moving the leading pebble along label 0 it is a simple matter to advance both pebbles one vertex while maintaining the invariant. A similar construction works with only one passive pebble, if the automaton can jump.

Cook and Rackoff [19, Theorem 4.14] present a family of 3-regular graphs that cannot be traversed using a constant number of states and pebbles, even if jumping is allowed and the edge labels are disclosed. The price paid to capture this strengthened model is a bound that is quantitatively weaker than that of Theorem 20. For instance, they do not rule out the combination $Q = O(1)$ and $P = O(\log\log n)$.

6.2. The Form of Universal Traversal Sequences

As another byproduct of Lemma 19, 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 of Lemma 19 shows that this is impossible.

Corollary 21: 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.

Proof: Since $|\alpha| < n/2$, the chain C_α of Lemma 19 can be embedded in a cycle of length n. Consider a traversal according to α starting at M. If $|\alpha|$, the distance from M to L, is even then, according to Lemma 19, the traversal ends at M. If $|\alpha|$ is odd then the traversal ends at a vertex an even distance from L, so that a second traversal according to α returns to M. In either case a traversal according to $\alpha\alpha$ starting at M returns to M after visiting at most $|\alpha| + 1 < n$ distinct vertices. Therefore α^k starting at M never visits more vertices. □

Using similar techniques, Theorem 22 proves that the previous result in fact holds for any even length α such that α is not a universal traversal sequence for all labeled $(n/2)$-cycles. For instance, it holds for any α whose length is even and $O(n^{1.29})$ (Tompa [36]).

Theorem 22: For any $\alpha \in \{0,1\}^*$ of even length, any even integer n, and any integer k, if α is not a universal traversal sequence for all labeled $(n/2)$-cycles, then α^k is not a universal traversal sequence for all labeled n-cycles.

Proof: Since α is not a universal traversal sequence for all labeled $(n/2)$-cycles, there is a labeled chain C of $n/2-1$ vertices with a vertex S such that starting at S and traversing according to α never leaves C and ends at some vertex T. Construct a cycle of length n as follows (see Figure 2): take a copy of C in which T is clockwise from S, followed by a new vertex M, followed by a copy C' of C in which the copy T' of T is counterclockwise from the copy S' of S, followed by a new vertex X.

Now start at any vertex s on the arc between S and S' containing M, where s is an even distance from S, and traverse according to α. This must terminate at a vertex t on the arc between T and T' containing M, where t is also an even distance from S, without ever reaching X. The reason t is between T and T' is that the walk from s to t is trapped between the walks from S to T and from S' to T'. The reason t is an even distance from S is because s is, and because $|\alpha|$ is even.

Therefore, starting at S and traversing according to α^k will never reach X, for any k. □

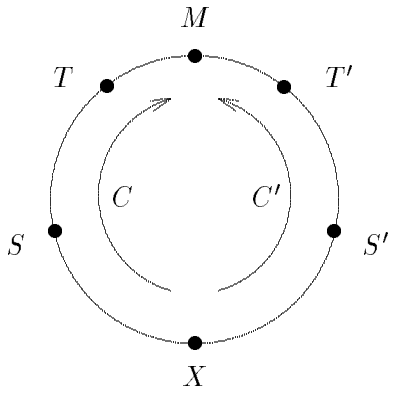


Figure 2: A Cycle Constructed from a Chain and its Reversal

7. 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 5 to bijectively labeled graphs, or to strengthen the bounds for automata having more pebbles than the graph's degree. Cook and Rackoff [19, 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 5.

References

[1] L. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science*, pages 75–83, Ann Arbor, MI, Oct. 1978. IEEE.

[2] 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.

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

[4] 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.

[5] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber. A sublinear space, polynomial time algorithm for directed *s-t* connectivity. In *Proceedings, Structure in Complexity Theory, Seventh Annual Conference*, pages 27–33, Boston, MA, June 1992. IEEE.

[6] G. Barnes and W. L. Ruzzo. Deterministic algorithms for undirected *s-t* connectivity using polynomial time and sublinear space. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 43–53, New Orleans, LA, May 1991.

[7] 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.

[8] 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.

[9] 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.

[10] B. Bollobás. *Extremal Graph Theory with Emphasis on Probabilistic Methods*, volume 62 of *Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences by the American Mathematical Society, 1986.

[11] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. MacMillan, 1976. Revised paperback edition, 1977.

[12] A. Borodin. Structured vs. general models in computational complexity. *L'Enseignement Mathématique*, XXVIII(3-4):171–190, July-Dec. 1982. Also in [29, pages 47–65].

[13] 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.

[14] A. Borodin, W. L. Ruzzo, and M. Tompa. Lower bounds on the length of universal traversal sequences. *Journal of Computer and System Sciences*, 45(2):180–203, Oct. 1992.

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

[16] 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.

[17] 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 Twenty First Annual ACM Symposium on Theory of Computing*, pages 574–586, Seattle, WA, May 1989.

[18] A. Cobham. The recognition problem for the set of perfect squares. Research Paper RC-1704, IBM Watson Research Center, 1966.

[19] 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.

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

[21] A. Hemmerling. *Labyrinth Problems: Labyrinth-Searching Abilities of Automata*, volume 114 of *Teubner-Texte zur Mathematik*. B. G. Teubner Verlagsgesellschaft, Leipzig, 1989.

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

[23] S. Hoory and A. Wigderson. Universal sequences for expander graphs. Hebrew University, Jerusalem, Dec. 1989.

[24] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, Oct. 1988.

[25] 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.

[26] S. Istrail. Constructing generalized universal traversing sequences of polynomial size for graphs with small diameter. In *31st Annual Symposium on Foundations of Computer Science*, pages 439–448, St. Louis, MO, Oct. 1990. IEEE.

[27] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks. On the cover time of random walks on graphs. *Journal of Theoretical Probability*, 2(1):121–128, Jan. 1989.

[28] 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.

[29] *Logic and Algorithmic,* An International Symposium Held in Honor of Ernst Specker, Zürich, Feb. 5–11, 1980. Monographie No. 30 de L'Enseignement Mathématique, Université de Genève, 1982.

[30] N. Nisan. $RL \subseteq SC$. In *Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing*, pages 619–623, Victoria, B.C., Canada, May 1992.

[31] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *33rd Annual Symposium on Foundations of Computer Science*, Pittsburgh, PA, Oct. 1992. IEEE.

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

[33] W. J. Savitch. Maze recognizing automata and nondeterministic tape complexity. *Journal of Computer and System Sciences*, 7(4):389–403, 1973.

[34] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[35] 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.

[36] M. Tompa. Lower bounds on universal traversal sequences for cycles and higher degree graphs. *SIAM Journal on Computing*, 21(6):1153–1160, Dec. 1992.

[37] D. Zuckerman. On the time to traverse all edges of a graph. *Information Processing Letters*, 38(6):335–337, 28 June 1991.

(Last RCS Revision: 1.126 (Long) Date: 1993/03/03 05:38:21 .)