The diameter of randomly perturbed digraphs and some applications

Abraham D. Flaxman*and Alan M. Frieze†
Department of Mathematical Sciences,
Carnegie Mellon University,
Pittsburgh, PA, 15213, USA
abie@cmu.edu, alan@random.math.cmu.edu

February 9, 2007

Abstract

The central observation of this paper is that if ϵn random arcs are added to any n-node strongly connected digraph with bounded degree then the resulting graph has diameter $\mathcal{O}(\ln n)$ with high probability. We apply this to smoothed analysis of algorithms and property testing.

Smoothed Analysis: Recognizing strongly connected digraphs is a basic computational task in graph theory. Even for digraphs with bounded degree, it is **NL**-complete. By XORing an arbitrary bounded degree digraph with a sparse random digraph $R \sim \mathbb{D}_{n,\epsilon/n}$ we obtain a "smoothed" instance. We show that, with high probability, a log-space algorithm will correctly determine if a smoothed instance is strongly connected. We also show that if **NL** $\not\subseteq$ almost-**L** then no heuristic can recognize similarly perturbed instances of (s,t)-connectivity.

Property Testing: A digraph is called k-linked if, for every choice of $2k$ distinct vertices $s_1, \ldots, s_k, t_1, \ldots, t_k$, the graph contains k vertex disjoint paths joining s_r to t_r for $r = 1, \ldots, k$. Recognizing k-linked digraphs is **NP**-complete for $k \geq 2$. We describe a polynomial time algorithm for bounded degree digraphs which accepts k-linked graphs with high probability, and rejects all graphs which are at least ϵn arcs away from being k-linked.

1 Introduction

The diameter of a graph G is the length of the longest shortest path in G. In other words, if $d(u,v)$ is the length of the shortest path from u to v in G, then the diameter of G is $\max_{u,v} d(u,v)$. A graph is connected (and a directed graph is strongly connected) if it has finite diameter. The central observation of this paper is that if ϵn random edges are added to any n-node connected graph with degree not-too-large then the diameter becomes $\mathcal{O}(\ln n)$ with high probability (throughout this paper, "with high probability" will mean with probability tending to 1 as $n \to \infty$ and will

*Supported in part by NSF Grant CCR-0122581
†Supported in part by NSF Grant CCR-0200945

be abbreviated **whp**). This is also true for strongly connected directed graphs and digraphs with not-too-large in-degree and out-degree and for several ways of generating the random edges. For ease of exposition, we state this as a theorem only for a strongly connected digraph $\bar{D}$ with degree $\mathcal{O}\left(n^{\epsilon/100}\right)$ that is perturbed by adding a random digraph $R \sim \mathbb{D}_{n,\epsilon/n}$. Here $R \sim \mathbb{D}$ means R is distributed according to distribution $\mathbb{D}$, and $\mathbb{D}_{n,p}$ is the distribution over of digraphs on vertex set $[n]$ in which each possible arc appears independently with probability p (so each digraph with m arcs is realized with probability $\binom{n(n-1)}{m} p^m (1-p)^{n(n-1)-m}$). Below, we use the notation $D = \bar{D} + R$ to mean that D is the graph formed by taking the union of the arcs of $\bar{D}$ and R.

Theorem 1 *Let ϵ be a positive constant with $\epsilon \leq 1$ and let $\Delta = n^{\epsilon/100}$. Let $\bar{D}$ be a strongly connected n-node digraph with in-degree and out-degree at most Δ. Let $D = \bar{D} + R$ where $R \sim \mathbb{D}_{n,\epsilon/n}$. Then* **whp** *the diameter of D is at most $100\epsilon^{-1} \ln n$.*

Similar results hold for perturbations formed by adding ϵn arcs selected at random with or without replacement, or by adding a random assignment with ϵn arcs.

Theorem 1 is related to a class of problems regarding the possible change of diameter in a graph where edges are added or deleted, for example, the results of Alon, Gryárfás, and Ruszinkó in [2] on the minimum number of edges that must be added to a graph to transform it into a graph of diameter at most d. The study of these extremal diameter alteration questions was initiated by Chung and Garey in [12]. It is also related to the theorem of Bollobás and Chung on the diameter of a cycle plus a random matching [7].

1.1 Application: Smoothed Analysis

A digraph is strongly connected if, for every vertex pair (s, t), there is a directed path from s to t. Recognizing strongly connected digraphs is a basic computational task, and the set of strongly connected digraphs is **NL**-complete [21]. Thus, if $\mathbf{NL} \not\subseteq \mathbf{L}$ then there is no log-space algorithm which recognizes strongly connected digraphs.

Perhaps this conclusion of worst-case complexity theory is too pessimistic. We will consider the performance of a simple heuristic which runs in randomized log-space. We will show that the heuristic succeeds on random instances **whp**. However, the "meaning" of this result depends on the probability space from which we draw the random instances. It seems reasonable to assume that most real-world digraphs will contain some amount of randomness, so it is tempting to believe this result shows that in the real-world strong connectivity only requires log-space. Unfortunately, this is not valid if we use the "wrong" model for randomness. For example, the distribution $\mathbb{D}_{n,p}$ (which is generated by taking n nodes and including each ordered pair as an arc independently with probability p) is pleasant for analysis, but basic statistics like the degree sequence seem to differ from several observed instances of real-world graphs [14].

We will use a model of randomness that is more flexible. We will start with an arbitrary digraph $\bar{D}$ and perturb it by XORing it with a very sparse random graph $R \sim \mathbb{D}_{n,\epsilon/n}$. This produces a random instance which is "less random" than $\mathbb{D}_{n,p}$. The study of worst case instances with small random perturbations is called Smoothed Analysis.

Smoothed Analysis was introduced by Spielman and Teng in [28] and they discuss a perturbation

model for discrete problems in [29]. They consider perturbing graphs by XORing the adjacency matrix with a random adjacency matrix, where each edge is flipped with some constant probability. Since the probability of an edge flip is constant, the perturbed instances are all dense graphs (i.e. a constant fraction of all possible edges appear). Independently, Bohman, Frieze and Martin [10] studied the issue of Hamiltonicity in a dense graph when random edges are *added*, and other graph properties were analyzed in this model by Bohman, Frieze, Krivelevich and Martin [11] and Krivelevich and Tetali [23].

We will also use an XOR perturbation, but we will make the probability of corruption much lower than [29]. Since we will have a linear number of arcs present, it is appropriate for the perturbation to change about ϵn arcs, which is the expected number of arcs in $\mathbb{D}_{n,\epsilon/n}$.

1.1.1 Randomness and strong connectivity

Recognizing strongly connected digraphs is closely related to recognizing (s,t)-connectivity in digraphs, which is the canonical **NL**-complete problem. It is possible to recognize connectivity in undirected graphs with a randomized log-space algorithm using random walks [1] (also, a deterministic log-space algorithm was recently discovered [25]). Since the cover time of an arbitrary connected graph is bounded by $\mathcal{O}(n^3)$ (see [17] for a sharp bound), a random walk will visit every vertex in polynomial time **whp**. This approach will not work for arbitrary digraphs, however, since there the cover time can be exponential.

The diameter and connectivity of random graphs has been well-studied, see for example the books of Bollobás [6] and Janson, Łuczak, and Ruciński [20]. Perhaps closest in spirit to our investigation is the paper of Bollobás and Chung on the diameter of a Hamilton cycle plus a random matching [7] and the paper of Chung and Garey on the diameter of altered graphs [12]. Also, the component structure of random digraphs was studied by Karp in [22] and more recently by Cooper and Frieze [13].

1.1.2 A heuristic for recognizing strong connectivity

For each ordered pair of vertices (s,t), repeat the following procedure N_1 times: starting from s, take N_2 steps in a random walk on the digraph. Here a random walk is the sequence of vertices $X_0, X_1, \ldots, X_t, \ldots$ visited by a particle which moves as follows: if $X_t = v$ then X_{t+1} is chosen uniformly at random from the out-neighbors of X_t. If any of the random walks ever reaches t then the digraph contains an (s,t)-path, and we continue to the next pair of vertices.

If N_1 and N_2 are large enough, this algorithm is correct **whp**. For example, if there is a path from s to t, then if the random walk has followed it correctly so far, it has probability more than $1/n$ of following it correctly for one more step. Since the distance from s to t is less than n, taking $N_2 = n$ we have that the success probability for a single walk exceeds n^{-n}. So taking $N_1 = n^{2n}$ we will discover the path **whp**. We have just given a superexponential time algorithm for a problem in **NL** but the values of N_1 and N_2 can be significantly improved for smoothed random instances.

The main theorem of this section is that when N_1 and N_2 are suitable functions of n, this heuristic, which we will call Algorithm $\mathcal{A}$, uses logarithmic space and is successful on perturbations of bounded

out-degree instances **whp**. To prove this, we first show that it is successful when the initial instance is a strongly connected digraph and the perturbation only adds arcs. Then we extend this to show success when the initial instance is not necessarily strongly connected and the perturbation only adds arcs. After this, it is simple to translate our results to the original perturbation model where arcs are added and removed, since we can generate the perturbation in 2 rounds, by first deleting each existing arc with some probability, and then adding random arcs to the resulting digraph.

Recall that $\mathbb{D}_{n,\epsilon/n}$ is the distribution of digraphs on vertex set $[n]$ in which each possible arc appears independently with probability ϵ/n, and we write $R \sim \mathbb{D}_{n,\epsilon/n}$ to mean R is selected randomly according to distribution $\mathbb{D}_{n,\epsilon/n}$. We write $G_1 \oplus G_2$ to mean the XOR of digraphs G_1 and G_2, (which is to say $e \in G_1 \oplus G_2$ if and only if $e \in G_1$ and $e \notin G_2$ or vice versa.)

Theorem 2 *Let ϵ and Δ be positive constants with ϵ sufficiently small. For any n-node digraph $\bar{D}$ with maximum in-degree and out-degree Δ, let $D = \bar{D} \oplus R$ where $R \sim \mathbb{D}_{n,\epsilon/n}$. Then there exist absolute constants A_1, B_1 such that* **whp** *Algorithm $\mathcal{A}$ is correct on D when $N_1 = n^{A_1 \epsilon^{-1} \ln(10\Delta)}$ and $N_2 = B_1 \epsilon^{-1} \ln n$.*

We find that $A_1 = 400$ and $A_2 = 200$ suffice in this theorem, but we do not attempt to optimize these values.

If a strongly connected digraph has bounded out-degree and has diameter $\mathcal{O}(\ln n)$ then a random walk of length $\mathcal{O}(\ln n)$ has a $1/\operatorname{poly}(n)$ chance of going from s to t, and Algorithm $\mathcal{A}$ will succeed **whp** using values of N_1 and N_2 that can be realized in log-space. Unfortunately, even though our initial instances have bounded out-degree and, as shown by Theorem 1, our perturbed instances have logarithmic diameter, the perturbation increases the maximum degree to $\Omega(\ln n / \ln \ln n)$, so we must work a little harder to show that the random walk has a non-negligible probability of witnessing the path from s to t. (As an additional reward for this work, we find that Algorithm $\mathcal{A}$ can be derandomized by checking all paths from s of length $\mathcal{O}(\epsilon^{-1} \ln n)$ and still only use log-space.)

The analysis of Algorithm $\mathcal{A}$ is further complicated by the possibility of an instance $\bar{D}$ which is not strongly connected combining with R to produce a smoothed instance which is strongly connected. We handle this situation by a three-step argument. First, for the smoothed instance to become strongly connected there cannot be too many small strongly connected components of $\bar{D}$. Then, the large components must merge to form a strongly connected component with low diameter. Finally, the small components, if they are connected to the large component, must be "close" to it.

1.1.3 Why study instances with bounded degree?

It would be nice to extend our results to hold for perturbed copies of any digraph, instead of only digraphs with bounded degree. However, such a result is not possible for our heuristic. We show that our assumption that $\bar{D}$ has bounded degree cannot be weakened too much by constructing a family of instances with maximum out-degree and maximum in-degree growing with n for which Algorithm $\mathcal{A}$ does not succeed **whp**.

Theorem 3 *Let ϵ be a sufficiently small positive constant, and let $R \sim \mathbb{D}_{n,\epsilon/n}$. Then, for every sufficiently large n, there exists an n-node digraph $\bar{D}$ with maximum out-degree $\mathcal{O}(\ln n)$ and max-*

imum in-degree $\mathcal{O}(n^{1/3}(\ln n)^2)$ *such that for* $\bar{D} \oplus R$ *the probability that Algorithm* $\mathcal{A}$ *fails exceeds* $e^{-\epsilon} - o(1)$.

1.1.4 Strong connectivity versus (s,t)-connectivity

Strong connectivity is an **NL**-complete problem, but (s,t)-connectivity is "the" **NL**-complete problem. In Sipser's undergraduate text [27], the completeness of (s,t)-connectivity is proved in detail, while the completeness of strong connectivity is left as an exercise (Appendix A includes a simple solution to this exercise which shows that completeness still persists for strong connectivity in graphs with bounded out-degree.)

In light of this, it is natural to investigate the success of heuristics on smoothed instances of (s,t)-connectivity. Here we find that there are instances on which Algorithm $\mathcal{A}$ fails **whp**. What is more, no log-space heuristic exists, provided a conjecture of complexity theory holds.

Theorem 4 *If* **NL** $\not\subseteq$ *almost-***L** *then no log-space heuristic succeeds* **whp** *on smoothed instances of bounded out-degree* (s,t)*-connectivity.*

The proof consists of building a machine which simulates any nondeterministic log-space machine using the log-space heuristic for (s,t)-connectivity, were such a heuristic to exist. Before the proof, we will also recall the definition of almost-**L** and comment on why it appears instead of **BPL**.

1.1.5 Smoothed model versus semi-random model

The semi-random model was introduced by Santha and Vazirani in [26]. In this model an adversary adaptively chooses a sequence of bits and each is corrupted independently with probability δ. The authors propose this as a model for real-world random bits, such as the output of a Geiger counter or noisy diode, and consider the possibility of using such random bits in computation on worst-case instances. Blum and Spencer consider the performance of a graph coloring heuristic on random and semi-random instances in [9]. Subsequent work has uncovered an interesting difference between the random and semi-random instances in graph coloring. The work of Alon and Kahale [3] developed a heuristic which succeeds **whp** on random instances with constant expected degree, while work by Feige and Kilian [16] showed no heuristic can succeed on semi-random instances with expected degree $(1-\epsilon)\ln n$ (they also developed a heuristic for semi-random instances with expected degree $(1+\epsilon)\ln n$).

In the original semi-random model of Santha and Vazirani, an instance is formed by an adaptive adversary, who looks at all the bits generated so far, asks for a particular value for the next bit, and gets the opposite of what was asked for with probability δ. Several modifications are proposed in Blum and Spencer [9] and also in Subramanian, Fürer, and Veni Madhavan [30] and Feige and Krauthgamer [15]. However, all these variations maintain the adaptive aspect of the adversary's strategy, which at low density allows too much power; if the error probability p equals $(1-\epsilon)\ln n/n$ then there will be roughly n^{ϵ} isolated vertices in $\mathbb{D}_{n,p}$ and the adversary can encode a polynomial-sized instance which contains no randomness. Since we wish to consider extremely sparse perturbations, where the error probability p equals ϵ/n, we cannot allow an adversary as

powerful as in the semi-random model. The XOR perturbation considered in this paper is equivalent to a natural weakening of the semi-random model: making the adversary oblivious.

1.2 Application: Property Testing

Property testing provides an alternative weakening of worst-case analysis of decision problems. It was formalized by Goldreich, Goldwasser, and Ron in [18]. The goal in property testing is to design an algorithm which decides whether an instance has a property or differs significantly from all instances which have that property (usually without looking at more than a vanishing fraction of the input bits). For example, a property tester for strong connectivity in bounded degree digraphs should accept all strongly connected instances and reject all instances that are ϵn arcs away from being strongly connected. Note that Algorithm $\mathcal{A}$ (which is designed to work on smoothed random instances) can be converted into a property tester: given an instance $\bar{D}$ and a gap parameter ϵ, we can randomly perturb $\bar{D}$ ourselves by adding $\frac{\epsilon}{2}n$ random arcs and then run Algorithm $\mathcal{A}$ on the perturbed version. This does not yield anything impressive for testing strong connectivity, since the undirected connectivity testing results of Goldreich and Ron in [19] can be applied to the directed case to produce a constant time tester. However, our perturbation approach also yields a property tester for a more difficult connectivity problem, that of being k-linked.

A digraph is said to be k-linked if for every choice of $2k$ distinct vertices $s_1, \ldots, s_k$, $t_1, \ldots, t_k$, the graph contains k vertex disjoint paths joining s_1 to t_1, $\ldots$, s_k to t_k. Recognizing whether or not a digraph is k-linked is **NP**-complete for $k \geq 2$. In the bounded-degree-property-testing version of being k-linked, we are given a constant ϵ and a digraph $\bar{D}$ with maximum in-degree and out-degree Δ and our goal is to accept if $\bar{D}$ is k-linked and reject if $\bar{D}$ is more than ϵn arcs away from being k-linked. (If $\bar{D}$ is not k-linked, but is close to being so, we can accept or reject it.)

1.2.1 A heuristic for testing k-linkedness

Given $\bar{D}$ and ϵ, we perturb $\bar{D}$ by generating a graph $R \sim \mathbb{D}_{n,\epsilon/2n}$ ourselves and adding that to D. Let $D = \bar{D} + R$ denote this perturbed instance. Then, for each choice of $2k$ distinct vertices, repeat the following procedure N_1 times: for $i = 1, \ldots, k$, starting at s_i take N_2 steps in a random walk on the graph. If all k random walks ever reach the correct k terminals via vertex disjoint paths, we continue to the next choice of $2k$ vertices. Otherwise reject.

Here k is assumed to be fixed, independent of the input.

Theorem 5 *Let ϵ and Δ be positive constants with ϵ sufficiently small. For any k-linked n-node graph $\bar{D}$ with maximum degree Δ, the algorithm above accepts in polynomial time* **whp**.

A few comments regarding the difference between this theorem and Theorems 1 and 2: the fact that k vertex disjoint paths exist **whp** follows from a calculation analogous to the proof of Theorem 1, but now we must explore disjoint neighborhoods around all $2k$ terminals simultaneously. Also, the analog of the most difficult part of Theorem 2, showing that Algorithm $\mathcal{A}$ is correct in the case where a disconnected $\bar{D}$ leads to a strongly connected D, is no longer necessary. In the property testing setting, we are not required to correctly recognize instances that lead to this situation. It

seems as though it might be possible to carry out this most difficult part and obtain a heuristic for testing k-linkedness that works on smoothed instances, but the details remain elusive.

Often in property testing, the goal is to minimize the sample complexity (meaning the number of times the algorithm accesses a bit of the input), and here we diverge from the norm, since the algorithm above likely looks at every arc of the graph. So it may be more accurate to call this an "algorithm for a promise problem version of k-linkedness". Also, we will not make use of the full power of ϵ-far-ness, and could succeed even on instances for which adding ϵn random arcs has probability less than $1/2$ of linking the unlinked.

1.3 Outline of what follows

We first prove Theorem 1 in Section 2. In Section 3 we prove Theorem 2, showing that Algorithm $\mathcal{A}$ is successful **whp**. Section 4 is devoted to the proof of Theorem 3, by constructing an instance with growing out-degree where Algorithm $\mathcal{A}$ fails with constant probability. In Section 5, we prove Theorem 4 by showing how to use a log-space heuristic for (s,t)-connectivity to build an almost-**L** simulator for any **NL** machine. Finally, in Section 6 we will prove Theorem 5, which is a reprise of the proof of Theorem 1. Section 7 is a brief conclusion.

1.4 Some facts and notation

We will use the following Chernoff bounds from [20, Theorem 2.1] on the Binomial random variable $\mathrm{B}(n,p)$:

$$\Pr\left[\mathrm{B}(n,p) \geq np + t\right] \leq \exp\left\{-\frac{t^2}{2(np+t/3)}\right\}, \tag{1}$$

$$\Pr\left[\mathrm{B}(n,p) \leq np - t\right] \leq \exp\left\{-\frac{t^2}{2np}\right\}. \tag{2}$$

$\mathbb{D}_{n,\epsilon/n}$ is the distribution of digraphs on vertex set $[n]$ in which each possible arc appears independently with probability ϵ/n, and we write $R \sim \mathbb{D}_{n,\epsilon/n}$ to mean R is selected randomly according to distribution $\mathbb{D}_{n,\epsilon/n}$.

We write $G_1 \oplus G_2$ to denote the digraph formed by XOR-ing digraphs G_1 and G_2, (which is to say $e \in G_1 \oplus G_2$ if and only if $e \in G_1$ and $e \notin G_2$ or vice versa.)

We write $G_1 + G_2$ to denote the digraph formed by taking the union of the arcs of G_1 and G_2.

2 Proof of Theorem 1

We now show that if $\bar{D}$ is strongly connected and has in-degree and out-degree bounded by $\Delta = \mathcal{O}(\ln n)$ then, for $R \sim \mathbb{D}_{n,\epsilon/n}$, the diameter of $D = \bar{D} + R$ is $\mathcal{O}(\epsilon^{-1} \ln n)$ **whp**.

We will show that **whp** D contains short paths of a special form, alternating between some arcs from $\bar{D}$ and random arcs from R. This is similar to the approach of Bollobás and Chung [7].

To proceed, we now fix 2 vertices s and t and look for a short path between them. Let P be a shortest path from s to t in $\bar{D}$. If P has length less than $100\epsilon^{-1}\ln n$ then this vertex pair is already close, so suppose P has length at least $100\epsilon^{-1}\ln n$.

Let S_0 be the first $32\epsilon^{-1}\ln n$ nodes of P and let T_0 be last $32\epsilon^{-1}\ln n$ nodes of P.

We call a node *useful* if it is not within distance $d = 5\epsilon^{-1}$ of any node which we have previously placed in any set S_i or T_i, where distance is the length of the shortest path in the undirected graph underlying $\bar{D}$.

To build S_i, for each node $s' \in S_{i-1}$, we check if R contains an arc from s' to some useful node s''. If it does, we include s'' in S_i and also all nodes reachable from s'' by taking d steps in $\bar{D}$.

T_j is defined analogously, but the paths lead towards t instead of away from s. So, for each $t' \in T_{i-1}$, if R contains an arc from some useful node t'' to t', we include t'' in T_j and also all nodes from which t'' is reachable by taking d steps in $\bar{D}$. To make this definition completely precise, we include a pseudocode description of the procedure which produces S_i and T_j, `GenerateSets`, in Figure 1. We use U to denote the set of useful nodes. Also, the notation $N_d^+(S)$ denotes the set of nodes reachable in $\bar{D}$ in at most d steps starting from some node of S, the notation $N_d^-(S)$ denotes the set of nodes from which some node of S is reachable in at most d steps in $\bar{D}$, and $N_d(S) = N_d^+(S) \cup N_d^-(S)$. Finally, let $\ell = \lceil \log_2 n \rceil$.

This procedure is convenient for analysis because no arc of R is examined more than once, due to the way the useful set U is maintained. Therefore, we can employ the *principle of deferred decisions* find a simple expression for the conditional probability that, for example, $(s', s'') \in R$ at any step of `GenerateSets`.

We will now show that when `GenerateSets` halts

$$\Pr[|S_i| \le n^{2/3} \text{ or } |T_j| \le n^{2/3}] = o(n^{-2}). \tag{3}$$

To see this, we first note that at any step of `GenerateSets`, $|U| \ge n - 2\Delta^{2d}(\ell n^{2/3} + 32\epsilon^{-1}\ln n) = (1 - o(1))n$. This is because at most $\Delta^{2d} \le n^{1/10}$ nodes are removed from U in any step where U is changed, and it is changed at most $n^{2/3}$ times in each inner loop, and the inner loops are executed at most ℓ times each. And, by similar considerations, the initialization of U has size at least $n - 2\Delta^{2d}(32\epsilon^{-1}\ln n)$.

Now we consider the event $\mathcal{E}_{s'}$ given by "$s' \in S_{i'}$ and there exists $s'' \in U$ with $(s', s'') \in R$." Since each arc appears in R independently with probability ϵ/n, we can apply the principle of deferred decisions. We condition on the entire history of `GenerateSets`, which can be described by $H = \langle U, S_1, T_1, \ldots, S_{i'}, T_{j'}, \tilde{S}_{i'+1}\rangle$, where $\tilde{S}_{i'+1}$ denotes the intermediate state of the set $S_{i'+1}$, and for $s' \in S_{i'}$, we have that the probability of $\mathcal{E}_{s'}$ depends only on the size of U, which is always $(1 - o(1))n$. So

$$\Pr[\mathcal{E}_{s'} \mid H] = 1 - (1-p)^{|U|} = (1 - o(1))\epsilon.$$

Every time $\mathcal{E}_{s'}$ occurs, at least d vertices are added to $S_{i'+1}$ (since $|N_d^+(s'')| \ge d$), so conditioned on $|S_{i'}|$, the random variable $|S_{i'+1}|/d$ stochastically dominates $Z_{i'+1} \sim \mathrm{B}(|S_{i'}|, (1 - o(1))\epsilon)$. Thus,

procedure `GenerateSets`$[(s,t)$-path $P]$

$S_0 :=$ first $32\epsilon^{-1}\ln n$ nodes of P.
$T_0 :=$ last $32\epsilon^{-1}\ln n$ nodes of P.
$U := V \setminus N_d(S_0 \cup T_0)$

$i := 0$
$j := 0$

while $(|S_i| \le n^{2/3}$ and $i \le \ell)$ or $(|T_j| \le n^{2/3}$ and $j \le \ell)$ **do**
  if $|S_i| \le n^{2/3}$ and $i \le \ell$ **then**
    $S_{i+1} := \emptyset$
    for all $s' \in S_i$ **do**
      if $|S_{i+1}| \le n^{2/3}$ and there exists $s'' \in U$ such that $(s', s'') \in R$ **then**
        $S_{i+1} := S_{i+1} \cup N_d^+(\{s''\})$
        $U := U \setminus N_d(S_{i+1})$
      end if
    end for
    $i := i + 1$
  end if

  if $|T_j| \le n^{2/3}$ and $j \le \ell$ **then**
    $T_{j+1} := \emptyset$
    for all $t' \in T_j$ **do**
      if $|T_{j+1}| \le n^{2/3}$ and there exists $t'' \in U$ such that $(t'', t') \in R$ **then**
        $T_{j+1} := T_{j+1} \cup N_d^-(\{t''\})$
        $U := U \setminus N_d(T_{j+1})$
      end if
    end for
    $j := j + 1$
  end if
end while

Figure 1: Pseudocode to generate S_i and T_j

letting $\mathcal{B}_{i'+1}$ denote the event "$|S_{i'+1}| \le 2|S_{i'}|$" we have

$$\Pr\left[\mathcal{B}_{i'+1} \;\middle|\; S_{i'}\right] \le \Pr\left[Z_{i'+1} \le \mathrm{E}\left[Z_{i'+1}\right] - \frac{3}{5}\epsilon|S_{i'}| \;\middle|\; S_{i'}\right] \le e^{-\frac{9}{50}\epsilon|S_{i'}|},$$

where the final inequality is an application of the Chernoff bound (2).

Note that in order for `GenerateSets` to halt with $|S_i| \le n^{2/3}$ it must be that some $\mathcal{B}_{i'}$ occurs for $i' \le i$. Since $|S_0| = 32\epsilon^{-1}\ln n$, we have that

$$\Pr[|S_i| \le n^{2/3}] \le \Pr\left[\bigcup_{i'=1}^{i} \mathcal{B}_{i'}\right] \le \sum_{i'=1}^{i} \Pr\left[\mathcal{B}_{i'} \mid |S_{i'-1}| \ge 32\epsilon^{-1}\ln n\right] \le \ell \cdot e^{-5\ln n} = o(n^{-2}).$$

A similar argument shows that when `GenerateSets` halts we also have $\Pr[|T_j| \le n^{2/3}] = o(n^{-2})$.

Now, to finish the short path from s to t, we generate the random arcs of R between S_i and T_j

$$\Pr\left[R \cap S_i \times T_j = \emptyset \;\middle|\; |S_i| \geq n^{2/3} \wedge |T_j| \geq n^{2/3}\right] \leq (1-p)^{n^{4/3}} \leq e^{-\epsilon n^{1/3}} = o(n^{-2}).$$

Putting all the pieces together, we have an (s,t)-path consisting of a path of length at most $32\epsilon^{-1}\ln n$, followed by at most 2ℓ paths in $\bar{D}$ of length at most $d+1$ joined by edges from R, and finishing with a path of length at most $32\epsilon^{-1}\ln n$, for total length which numerical calculation shows is less than $100\epsilon^{-1}\ln n$.

Since there are only $n(n-1)$ choices for (s,t), the theorem follows by the union bound. □

3 Proof of Theorem 2

3.1 When $\bar{D}$ is strongly connected and perturbation does not delete edges

By Theorem 1 we know that the diameter of D is $\mathcal{O}(\ln n)$ **whp**. Unfortunately we cannot yet conclude that Algorithm $\mathcal{A}$ is successful **whp**. We must still argue that the probability of a random walk traversing the short path is not too small. In a graph with out-degree less than some constant, having a diameter of $\mathcal{O}(\ln n)$ would imply an efficient algorithm. Our random perturbation has likely created some vertices with out-degree $\Omega(\ln n/\ln\ln n)$, so we will have to work a little more. We use the notation $\deg^+_D(v)$ to denote the out-degree of a vertex v in digraph D.

Lemma 6 *Let $D = \bar{D} + R$, where $\bar{D}$ is an arbitrary digraph with maximum out-degree Δ and $R \sim \mathbb{D}_{n,\epsilon/n}$. Then* **whp** *$D$ contains no path P of length $\ell \leq \ell_1 = 100\epsilon^{-1}\ln n$ with $\prod_{x\in P}\deg^+_D(x) \geq n^{100\epsilon^{-1}\ln(10\Delta)}$.*

Proof We prove the claim by the first moment method. First, we bound the number of paths with ℓ vertices that use $\ell - a$ arcs of $\bar{D}$. There are n places to start such a path, and there are $\binom{\ell}{a}$ different ways to decide when to take an arc not in $\bar{D}$. For each arc in $\bar{D}$, since the out-degree is bounded, there are at most Δ choices, and there are at most n^a choices for where the non-$\bar{D}$ arcs can go. So there are at most

$$n\Delta^{\ell-a}\binom{\ell}{a}n^a$$

potential paths of length ℓ that use a arcs from R. The probability such a potential path appears as a path in D is $\left(\frac{\epsilon}{n}\right)^a$.

Now we bound the probability that the sum of the logarithms of the out-degrees of the vertices along a path P of length ℓ exceeds $\ell\ln(\Delta+1)+t$. To do so, we first bound a similar quantity for the graph $R' = R \setminus P$. Note that

$$\Pr\left[\sum_{v\in P}\ln\{1+\deg^+_{R'}(v)\} \geq t\right] \leq e^{-t}\,\mathrm{E}\left[\prod_{v\in P}(1+\deg^+_{R'}(v))\right]$$
$$= e^{-t}\prod_{v\in P}\mathrm{E}\left[1+\deg^+_{R'}(v)\right] \leq (1+\epsilon)^{|P|}e^{-t}.$$

Then, since P is a path, it contains at most one arc incident with v, so

$$\ln\{\deg^+_D(v)\} \le \ln\{\deg^+_{\bar D}(v) + \deg^+_R(v)\} \le \ln\{\Delta + 1 + \deg^+_{R'}(v)\} \le \ln\{1+\Delta\} + \ln\{1+\deg^+_{R'}(v)\},$$

and we have

$$\Pr\left[\sum_{v\in P} \ln\{\deg^+_D(v)\} \ge \ell\ln(1+\Delta) + t\right] \le (1+\epsilon)^\ell e^{-t}.$$

So the expected number of paths of length ℓ with a arcs from R and product of degrees exceeding $\ell\ln(\Delta+1)+t$ is at most

$$n\Delta^{\ell-a}\binom{\ell}{a}n^a\left(\frac{\epsilon}{n}\right)^a(1+\epsilon)^\ell e^{-t} \le n((1+\epsilon)\Delta)^\ell e^{-t}\binom{\ell}{a}.$$

Let $\ell_1 = 100\epsilon^{-1}\ln n$ and let

$$t = 2\ln n + \ln\ell_1 + \ell_1\ln(2(1+\epsilon)\Delta) \le 100\epsilon^{-1}\ln(10\Delta)\ln n.$$

Then an upper bound on the probability that D contains such a path of length at most ℓ_1 is

$$\sum_{\ell=1}^{\ell_1}\sum_{a=0}^{\ell} ne^{-t}((1+\epsilon)\Delta)^\ell\binom{\ell}{a} = \sum_{\ell=1}^{\ell_1} ne^{-t}((1+\epsilon)\Delta)^\ell 2^\ell$$
$$= ne^{-t}\sum_{\ell=1}^{\ell_1}(2(1+\epsilon)\Delta)^\ell \le ne^{-t}\ell_1(2(1+\epsilon)\Delta)^{\ell_1}\} = o(1).$$

So **whp** there is no path P of length at most $100\epsilon^{-1}\ln n$ which has

$$\prod_{x\in P}\deg^+_D(x) \ge n^{100\epsilon^{-1}\ln(10\Delta)}.$$

□

The correctness of Algorithm $\mathcal{A}$ in the case when $\bar D$ is strongly connected now follows from the fact that the probability that a random walk follows a path P from s to t is precisely $\left(\prod_{x\in P}\deg^+_D(x)\right)^{-1}$.
□

3.2 When $\bar D$ not strongly connected and perturbation does not delete edges

The previous section shows that Algorithm $\mathcal{A}$ is correct **whp** for strongly connected digraphs $\bar D$. To prove that Algorithm $\mathcal{A}$ is correct **whp** when $\bar D$ is not strongly connected, we must do some more work.

Outline of approach
Consider the strong components of $\bar D$. If there are many components of size less than $\frac{1}{4}\epsilon^{-1}\ln n$, then we show that **whp** one of them will be incident to no arcs of R and so D will not be strongly connected and Algorithm $\mathcal{A}$ will be correct. In the case where $\bar D$ consists mostly of larger

strong components, we expose the random arcs R in two rounds. We argue that **whp** the strong components of $\bar{D}$ merge into a unique giant strong component S_g containing at least $n - n^{16/17}$ vertices after the first round. Then we invoke Lemma 1 from the previous section to show that the random arcs from the second round give the giant component a low diameter. Then we deal with the vertices that belong to small strong components after the first round. These vertices might be connected to S_g in both directions and they might not, and there is not necessarily a sharp threshold for strong connectivity. However, we show that, for some constant A_1, **whp** no vertex outside of S_g is connected to S_g only by paths of length more than $A_1\epsilon^{-1}\ln n$. This implies that any such vertex is close to the giant component, or cannot be reached at all. Finally, by Lemma 6, we know that all the paths of length at most $A_1\epsilon^{-1}\ln n$ have a non-negligible probability of being traversed by Algorithm $\mathcal{A}$ (take the bound in Lemma 6 and raise it to the power $A_1/100$). So we conclude that **whp** either the graph is not strongly connected, in which case Algorithm $\mathcal{A}$ is correct, or the graph *is* strongly connected, in such a way that Algorithm $\mathcal{A}$ is still correct.

The calculations required for this plan follow.

Lemma 7 *If $\bar{D}$ has more than $n^{1/2}\ln n$ strong components containing less than $\frac{1}{4}\epsilon^{-1}\ln n$ vertices, then* **whp** *one of these components is not incident to any arcs of R.*

Proof We use the second moment method (see, for example, [20, Page 54]). For each small strong component C of $\bar{D}$, let X_C be an indicator random variable for the event that C is not incident to any arc of R. Then $\mathrm{E}[X_C] = (1-\epsilon/n)^{2c(n-c)} \geq n^{-1/2}(1-o(1))$, where $c = |C| \leq \frac{1}{4}\epsilon^{-1}\ln n$, and

$$\begin{aligned}\mathrm{E}[X_{C_1}X_{C_2}] &= (1-\epsilon/n)^{2c_1(n-c_1-c_2)+2c_2(n-c_1-c_2)+2c_1c_2}\\ &= (1-\epsilon/n)^{2c_1(n-c_1)}(1-\epsilon/n)^{2c_2(n-c_2)}(1+\mathcal{O}((\ln n)^2/n))\\ &= \mathrm{E}[X_{C_1}]\,\mathrm{E}[X_{C_2}]\,(1+o(1)),\end{aligned}$$

where $c_1 = |C_1|$ and $c_2 = |C_2|$. Let $Z = \sum_C X_C$ be the number of strong components that are incident to no arc of R. Then, since there are at least $n^{1/2}\ln n$ terms in this sum, $\mathrm{E}[Z] \geq \frac{1}{2}\ln n$. We also have

$$\begin{aligned}\frac{\mathrm{E}\left[Z^2\right]}{\mathrm{E}[Z]^2} &= \frac{\sum_C \mathrm{E}\left[X_C^2\right] + \sum_{C_1\neq C_2}\mathrm{E}[X_{C_1}X_{C_2}]}{\left(\sum_C \mathrm{E}[X_C]\right)^2}\\ &= \frac{\sum_C \mathrm{E}[X_C]}{\left(\sum_C \mathrm{E}[X_C]\right)^2} + (1+o(1))\frac{\sum_{C_1\neq C_2}\mathrm{E}[X_{C_1}]\,\mathrm{E}[X_{C_2}]}{\left(\sum_C \mathrm{E}[X_C]\right)^2} \leq \frac{2}{\ln n} + 1 + o(1).\end{aligned}$$

Now, by the second moment method, (see, for example, [20, Inequality 3.3, Page 54]) we have $\Pr[Z \neq 0] \geq \mathrm{E}[Z]^2/\mathrm{E}\left[Z^2\right] = 1 - o(1)$. □

It follows from this lemma that Algorithm $\mathcal{A}$ works in the case where the number of small strong components of $\bar{D}$ is large.

We now consider the case where $\bar{D}$ has at most $n^{1/2}\ln n$ strong components of size less than $\sigma = \frac{1}{4}\epsilon^{-1}\ln n$. We consider adding the random arcs of R in two rounds, by introducing R' and R''. We take $R' \sim \mathbb{D}_{n,p'}$ with $p' = \frac{\epsilon}{2n}$ and $R'' \sim \mathbb{D}_{n,p''}$ with $p'' = \frac{\epsilon}{2n-\epsilon} = (1+o(1))\frac{\epsilon}{2n}$. Then the probability that an arc appears in $R'+R''$ is exactly $\frac{\epsilon}{n}$, and $R'+R''$ is identically distributed with R.

Let the strong components in $\bar{D}$ with size exceeding σ be $C_1, C_2, \ldots, C_a$ and let their sizes be $n_1, n_2, \ldots, n_a$. For $K \subseteq [a]$ let $C_K = \bigcup_{i\in K} C_i$, let $n_K = \sum_{i\in K} n_i$, and let A_K^+ denote the number of arcs of R' that go from C_K to $\overline{C_K}$ and let A_K^- denote the number of arcs of R' that go from $\overline{C_K}$ to C_K. Then

$$\Pr(A_K^+ = 0 \text{ or } A_K^- = 0) \le 2\left(1-\frac{\epsilon}{2n}\right)^{n_K(n-\sigma n^{1/2}\log n - n_K)}.$$

To obtain an upper bound on the number of choices for K with a given value of n_K, first note that, since each large strong component is of size at least σ, we have that a, the number of large strong components, is at most n/σ. Also, as a consequence of the strong components being large, for a given value of n_K, we have $|K| \le n_K/\sigma$. So the number of choices for K with a given value of n_K is at most $\sum_{\ell=1}^{n_K/\sigma}\binom{a}{\ell}$ and, for $n_K \le n/2$, this is at most $n_K/\sigma\binom{n/\sigma}{n_K/\sigma}$. Thus

$$\begin{aligned}
&\Pr(\exists K\subseteq[a]:\ n^{16/17} \le n_K \le n/2 \text{ and } A_K^+ = 0 \text{ or } A_K^- = 0)\\
&\quad\le \sum_{n_K=n^{16/17}}^{n/2} n_K/\sigma\binom{n/\sigma}{n_K/\sigma}2\left(1-\frac{\epsilon}{2n}\right)^{n_K(n(1-o(1))-n_K)}\\
&\quad\le \sum_{n_K=n^{16/17}}^{n/2} 2n_K/\sigma\left((ne/n_K)^{1/\sigma}e^{-\epsilon/4(1-o(1))}\right)^{n_K}\\
&\quad\le \sum_{n_K=n^{16/17}}^{n/2} e^{-(1-o(1))\epsilon n_K/68} = o(1).
\end{aligned}$$

It follows that **whp**

$$\bar{D}+R' \text{ contains a } \textit{giant} \text{ strong component } S_g \text{ with } |S_g| \ge n-(1+o(1))n^{16/17}. \tag{4}$$

We apply the results of the previous section to S_g. We have a strongly connected digraph S_g and we add $R'' \sim \mathbb{D}_{n,p''}$ (recall that $p'' = \epsilon/(2n-\epsilon)$), producing a digraph D_g with diameter at most $201\epsilon^{-1}\ln n$ for which all shortest paths P satisfy $\prod_{x\in P}\deg^+_{D_g}(x) \le n^{201\epsilon^{-1}\ln(10\Delta)}$.

The only detail remaining is how to deal with the at most $(1+o(1))n^{16/17}+\sigma n^{1/2}\ln n$ vertices of $\bar{D}+R'$ that are not in S_g. Let x be such a vertex. We will show that **whp** if there is a path from x to any vertex in S_g then it is a short path. An identical argument shows the same property holds for paths from S_g to x.

We consider two cases. Let V_x denote the set of vertices reachable by following $5\epsilon^{-1}\ln n$ arcs of $\bar{D}+R'$, starting from x. If $|V_x| \ge 5\epsilon^{-1}\ln n$ we say x is *medium* and if $|V_x| < 5\epsilon^{-1}\ln n$ we say x is *small*. If x is a medium vertex, then the probability R'' does not add an arc from a vertex in V_x to the S_g is at most $(1-p'')^{5\epsilon^{-1}\ln n(n-2n^{16/17})} \le n^{-2}$. Thus **whp** all medium x are close to S_g in D. Let S_m denote the set of medium vertices.

Consider now a shortest path in D from a small vertex x to a vertex z in $S_g \cup S_m$. Removing all of the arcs of R'' from this path decomposes it into subpaths $P_1, P_2, \ldots, P_r$. Let x_i and y_i denote the starting and ending vertices of subpath P_i (it is possible that $x_i = y_i$ if a subpath has length 0). We will show that **whp** r is small by considering the probability that the subpaths has a sequence $x_1, y_1, \ldots, x_r, y_r$ with $r \geq 18$.

For any $\bar{D} + R'$ for which (4) holds, the number of choices for our sequence is at most $(2n^{16/17} \times 5\epsilon^{-1} \ln n)^r n$ and the probability that the R''-arcs exist is $(p'')^r$. Thus the probability there exists a small vertex x which requires $r \geq 18$ is at most

$$\sum_{r \geq 18} n(2n^{16/17} \times 5\epsilon^{-1} \ln n \times p'')^r = o(1).$$

So **whp**, if D is strongly connected then its diameter is at most $201\epsilon^{-1} \ln n + 2 \times 18(5\epsilon^{-1} \ln n + 1)$ which, for n sufficiently large, is at most $400\epsilon^{-1} \ln n$. (The worst case here comes from a path of length at most $18(5\epsilon^{-1} \ln n + 1)$ from s to S_g, followed by a path of length at most $201\epsilon^{-1} \ln n$ to some other vertex of S_g and then by a path of length at most $18(5\epsilon^{-1} \ln n + 1)$ from there to t.)

By applying Lemma 6, we see that **whp** these paths are traversed with probability $n^{-400\epsilon^{-1} \ln(10\Delta)}$ and so Algorithm $\mathcal{A}$ is correct **whp**. □

4 Proof of Theorem 3: An example with growing degrees

We now consider the possibility of applying Algorithm $\mathcal{A}$ to the case where $\bar{D}$ has maximum in-degree and out-degree Δ that grows with n. The following example shows that in this case Algorithm $\mathcal{A}$ does not necessarily succeed when using logarithmic space.

Let $d = n^{1/3} \ln n$. To form our instance, we start with the directed cycle $C = (v_1, v_2 \ldots, v_n)$. Then, for each v_i with $i \geq n/d$, we let $i_0 = 1 + \lfloor (i - n/d)/d \rfloor$, and we add arcs (v_i, v_j) to $\bar{D}$, for each $j \in \{i_0, i_0 + 1, \ldots, i_0 + \ln n - 1\}$. We call these arcs the "backwards arcs". Note that each v_i with $i \geq n/d$ has $\ln n$ backwards arcs going out, and every v_j with $j < n/d$ has $d \ln n$ backwards arcs coming in. Let $V_1 = \{v_1, \ldots, v_{n/d + \ln n}\}$, and note that all backwards arcs lead to V_1.

When we perturb this $\bar{D}$ to obtain D, the probability that we do not delete any arc of cycle C is $(1 - \epsilon/n)^n = e^{-\epsilon} - o(1)$, and so D is strongly connected with probability at least $e^{-\epsilon} - o(1)$.

We now derive bounds showing two properties that hold **whp** and allow us to reason about the probability of Algorithm $\mathcal{A}$ succeeding on D.

Lemma 8 *Let $L = (3\epsilon)^{-1} \ln n$ and V_1 be as above. Then the following holds* **whp**

(a) *For $\ell \leq L$, there is no path from V_1 to $\{v_{n-L}, \ldots, v_n\}$ of length ℓ which does not use any of the backwards arcs.*

(b) *For $\ell > L$, there are at most $e^{2\epsilon\ell}$ paths of length ℓ from V_1 to v_n which do not use any of the backwards arcs.*

Proof Let P be a path from v_i to v_n of length at most ℓ. Removing all of the arcs of R from this path decomposes it into subpaths $P_1, \ldots, P_m$, where each P_j is a path of $\bar{D}$ (or possibly a degenerate path containing no arcs). Let the lengths of subpath P_j be denoted by $\ell_j \geq 0$ and let $\lambda = \ell_1 + \ldots + \ell_m$. Since the length of P is at most ℓ, we have $m \leq \ell$, and for a given m, we have $\lambda \leq \ell - m$. Also, P_1 starts at v_i and, since P_m ends at v_n, it must start at $v_{n-\ell_m}$. There are less than n possibilities for where the subpaths P_k begin for $k = 2, \ldots, m-1$. So, since there are $m-1$ arcs of R in P, each of which appears with probability ϵ/n, the expected number of path from v_i to v_n with length at most ℓ is at most

$$\begin{aligned}
\mathrm{E}\left[\#\text{ paths from } v_i \text{ to } v_n \text{ with length} \leq \ell\right] &\leq \sum_{m=0}^{\ell} \sum_{\lambda=0}^{\ell-m} \sum_{\ell_1+\cdots+\ell_m=\lambda} n^{m-2} \left(\frac{\epsilon}{n}\right)^{m-1} \\
&= n^{-1} \sum_{m=0}^{\ell} \epsilon^{m-1} \sum_{\lambda=0}^{\ell-m} \binom{\lambda+m-1}{m-1} \\
&= n^{-1} \sum_{m=0}^{\ell} \epsilon^{m-1} \binom{\ell}{m} \\
&\leq (\epsilon n)^{-1} \sum_{m=0}^{\ell} \frac{(\epsilon\ell)^m}{m!} \\
&\leq (\epsilon n)^{-1} e^{\epsilon\ell}.
\end{aligned}$$

(a) Since there are $(1+o(1))n/d$ vertices v_i in V_1, the probability that D contains a path of length at most L from some vertex of V_1 to v_n is at most

$$(1+o(1))(n/d)(\epsilon n)^{-1} e^{\epsilon L} = (1+o(1))d^{-1}\epsilon^{-1} n^{1/3} = o(1).$$

(b) Since there are $(1+o(1))n/d$ vertices v_i in V_1, the expected number of paths of length at most ℓ from some vertex of V_1 to v_n is at most $(1+o(1))(d\epsilon)^{-1}e^{\epsilon\ell}$ and then for $\ell \geq L$, the Markov inequality and the union bound show that

$$\Pr\left[\exists \ell \geq L : \text{ number paths } \geq e^{2\epsilon\ell}\right] \leq (1+o(1))(d\epsilon)^{-1} \sum_{\ell \geq L} e^{-\epsilon\ell} = o(1).$$

□

Now consider a random walk W from v_1 to v_n. Let the parts of W between the use of backwards arcs of $\bar{D}$ be called *attempts* and let W' denote the last attempt of W (denoted *successful*). All other attempts will be termed *failures*. Note that W' starts in V_1.

At some point Algorithm $\mathcal{A}$ checks for (v_1, v_n)-connectivity by executing T steps of a random walk from v_1 and declaring connectivity if v_n is reached. Also, this is to be repeated N times. Now we must have $T = n^{\mathcal{O}(1)}, N = n^{\mathcal{O}(1)}$ in order that we can realize the counters in log-space. The probability that any walk reaches v_n can be bounded by $T\sum_{\ell=L}^{n}(\ln n - 2)^{-\ell} e^{2\epsilon\ell} \leq (\ln n)^{-L/2}$. The factor T accounts for the $\leq T$ times at which the successful attempt may begin, $e^{2\epsilon\ell}$ bounds the number of possible paths making up this attempt and $(\ln n - 2)^{-\ell}$ bounds the probability we follow this path. This is because every vertex visited after i_0 on the successful attempt has out-degree at least $\ln n - 2$ **whp**. (Note that we may have deleted some of the backwards arcs when we XORed

with R, but the probability that there exists some vertex for which 2 backwards arcs are deleted is at most $n\binom{\ln n}{2}\left(\frac{\epsilon}{n}\right)^2 = o(1)$.)

Thus the probability that we will declare v_1 connected to v_n is at most $N(\ln n)^{-L/2} = o(1)$ and the algorithm fails with probability at least $e^{-\epsilon} - o(1)$.

(This constructions works with any slowly growing function as the out-degree, instead of $\ln n$. But to reduce the in-degree, it seems that some additional work is necessary.)

5 Proof of Theorem 4

Suppose a heuristic exists which uses log-space and is successful **whp** on smooth instances of (s,t)-connectivity. Then, using a log-space transducer, we convert a worst-case instance of (s,t)-connectivity on n nodes into a smoothed instance of (s,t)-connectivity. Unfortunately, this reduction is not sufficient to show the existence of a heuristic implies $\mathbf{NL} \subseteq \mathbf{BPL}$, since a nondeterministic log-space simulator does not have the space to store the output of the log-space transducer. The traditional technique for simulating a log-space machine on the output of a log-space transducer is to, each time a bit of the input is requested, restart the transducer and simulate it until it produces the bit in question. This is an inefficient use of time, but in exchange for taking longer we use only logarithmically bounded space.

In our case, since the reduction is randomized, we somehow need the log-space transducer to produce the *same* random instance each time. This seems to require "multiple access randomness" (also known as the "wrong" definition of **BPL** and denoted $\mathbf{BP}^\star\mathbf{L}$). Nisan provides some evidence that multiple access randomness is more powerful than read-once randomness in [24]. He also shows that $\mathbf{BP}^\star\mathbf{L} = \text{almost-}\mathbf{L} \subseteq \mathbf{L/poly}$ (where almost-**L** is the set of languages L for which $\mu(L \in \mathbf{L}^A) = 1$, where μ is the standard measure for the set of oracles. For more details on almost classes, see the survey of Vollmer and Wagner [31].)

Given an instance D_0 of (s,t)-connectivity on n nodes, we construct an instance $\bar{D}$ on n^3 nodes by adding $n^3 - n$ isolated vertices. Call the original n vertices A and the additional vertices B. We smooth the instance by XORing it with $R \sim \mathbb{D}_{n^3,\epsilon/n^3}$ to form $D = \bar{D} \oplus R$. (This is where our log-space machine uses multiple access randomness; there is not room to write out this whole graph, so we must generate the i-th bit from scratch every time the heuristic asks for it. The values of D differ from $\bar{D}$ with some small probability and they should differ in the same way every time the heuristic asks for the i-th bit.)

The probability R contains an arc between any pair of vertices of A is bounded by $n^2(\epsilon/n^3) = o(1)$. So if D_0 is (s,t)-connected, then D is (s,t)-connected **whp**. Now, since the vertices of B are isolated in $\bar{D}$, they form a sparse random graph in D, so **whp** no component has size exceeding $\mathcal{O}(\ln n)$ (see, for example, [20, Theorem 5.4]). Thus, the probability that D contains a component of B with arcs to and from vertices of A is less than $\mathcal{O}((n^3 \ln n)n^2(\epsilon/n^3)^2) = o(1)$. (Explanation: there are $n^3 - n < n^3$ choices for one endpoint in B, and $\mathcal{O}(\ln n)$ choices for the other, since it must be in the same component. There are $\mathcal{O}(n)$ choices for each endpoint in A, and the probability that each random arc appears is ϵ/n^3.)

Since, **whp**, there are no arcs added to A and no components of B that serve as a shortcut between

vertices of A, if D_0 is not (s,t)-connected, then D is not (s,t)-connected **whp**. This is sufficient to conclude that if a heuristic exists then $\mathbf{NL} \subseteq$ almost-$\mathbf{L}$, since, given any D_0, we could form D and use the heuristic to solve it, which would give the correct answer to the original problem about D_0 **whp**.

6 Proof of Theorem 5

We will show that if $\bar{D}$ is k-linked then **whp** D contains disjoint paths of length at most $100k\epsilon^{-1}\ln n$ which witness this.

Fix $s_1,\ldots,s_k,t_1,\ldots,t_k$, and let $P_1,P_2,\ldots,P_k$ be vertex disjoint paths in $\bar{D}$ such that P_r goes from s_r to t_r.

We order the paths from longest to shortest and define $r^\star$ so that, for $r \le r^\star$, each P_r has length at least $64k\epsilon^{-1}\ln n$. If $r^\star = 0$ then there is nothing to prove, so suppose $r^\star \ge 1$.

We use the same type of argument as in the proof of Theorem 1 to show the existence of short paths between s_r and t_r, but we work with all $r \le r^\star$ simultaneously to ensure that the paths that we find are vertex disjoint. To this end, we define a sequence of collections of subsets of vertices $S_{i,r}$ and $T_{i,r}$ for $i \ge 0$ and $1 \le r \le r^\star$, (we also define $S_{i,r} = P_i$ for $i \ge 0$ and $r > \ell$). Let $S_{0,r}$ be the first $32k\epsilon^{-1}\ln n$ vertices of P_r and $T_{0,r}$ be the last $32k\epsilon^{-1}\ln n$ vertices of P_r, for $1 \le r \le \ell$.

We will call a node *useful* if it is not within distance $d = 5\epsilon^{-1}$ of any node which we have previously placed in any S or T set, where "distance" is the length of the shortest path in the undirected graph corresponding to $\bar{D}$.

To define $S_{i,r}$, we check, for each node s' in $S_{i-1,r}$, if R contains an arc from s' to some useful node s''. If it does, we add s'' and all nodes reachable from s'' by d steps in $\bar{D}$ to $S_{i,r}$. Note that if s'' is useful, this will add at least d nodes to $S_{i,r}$.

$T_{j,r}$ is defined analogously, but the paths lead towards t_r instead of away from s_r. For a node t' in $T_{j-1,r}$, we look for useful nodes t'' where an arc of R is directed from t'' to t', and add all nodes from which t' is reachable by d steps in $\bar{D}$.

To make this definition completely precise, we include a pseudocode description of the procedure `GenerateSets2` for forming $S_{i,r}$ and $T_{j,r}$ in Figure 2. We use U to denote the set of useful nodes. Also, the notation $N_d^+(S)$ denotes the set of nodes reachable in $\bar{D}$ in at most d steps starting from some node of S, the notation $N_d^-(S)$ denotes the set of nodes from which some node of S is reachable in at most d steps in $\bar{D}$, and $N_d(S) = N_d^+(S) \cup N_d^-(S)$. Finally, let $\ell = \lceil \log_2 n \rceil$.

The proof is largely the same at Theorem 2. We now must argue that when `GenerateSets2` halts, $\Pr[S_{i_r,r} \le n^{2/3} \vee T_{j_r,r} \le n^{2/3}] = o(n^{-2})$.

As with `GenerateSets`, the procedure `GenerateSets2` is convenient for analysis because no arc of R is examined more than once, due to the way the useful set U is maintained. Therefore, we can employ the principle of deferred decisions find a simple expression for the conditional probability that, for example, $(s',s'') \in R$ at any step of the procedure.

We first note that at any step of `GenerateSets2`, $|U| \ge n - 2k\Delta^{2d}(\ell n^{2/3} + 32\epsilon^{-1}\ln n) = (1-o(1))n$.

This is because at most Δ^{2d} nodes are removed from U in any step where U is changed, and it is changed at most $n^{2/3}$ times in each inner loop, and the 2 inner loops are executed at most ℓk times each. And, by similar considerations, the initialization of U has size at least $n - 2k\Delta^{2d}(32\epsilon^{-1}\ln n)$.

Now we consider the event $\mathcal{E}_{s'}$ given by "$s' \in S_{i',r}$ and there exists $s'' \in U$ with $(s', s'') \in R$." Since each arc appears in R independently with probability ϵ'/n, we can apply the principle of deferred decisions. We condition on the entire history of the procedure, and for $s' \in S_{i',r}$, we have that the probability of $\mathcal{E}_{s'}$ depends only on the size of U, which is always $(1 - o(1))n$. So

$$\Pr[\mathcal{E}_{s'} \mid H] = 1 - (1-p)^{|U|} = (1 - o(1))\epsilon.$$

Every time $\mathcal{E}_{s'}$ occurs, at least d vertices are added to $S_{i'+1,r}$, so conditioned on $|S_{i',r}|$, the random variable $|S_{i'+1,r}|/d$ stochastically dominates $Z_{i'+1} \sim \mathrm{B}(|S_{i',r}|, (1-o(1))\epsilon)$. Thus, letting $\mathcal{B}_{i'+1}$ denote the event "$|S_{i'+1,r}| \le 2|S_{i',r}|$" we have

$$\Pr\left[\mathcal{B}_{i'+1} \;\middle|\; S_{i',r}\right] \le \Pr\left[Z_{i'+1} \le \mathrm{E}\left[Z_{i'+1}\right] - \frac{3}{5}\epsilon|S_{i',r}| \;\middle|\; S_{i',r}\right] \le e^{-\frac{9}{50}\epsilon|S_{i',r}|},$$

where the final inequality is an application of the Chernoff bound in (2).

Note that, in order for the procedure to halt with $|S_{i_r,r}| \le n^{2/3}$, it must be that some $\mathcal{B}_{i'}$ occurs for $i' \le i$. Since $|S_{0,r}| = 32\epsilon^{-1}\ln n$, we have that

$$\Pr\left[|S_{i_r,r}| \le n^{2/3}\right] \le \Pr\left[\bigcup_{i'=1}^{i_r} \mathcal{B}_{i'}\right] \le \sum_{i'=1}^{i_r} \Pr\left[\mathcal{B}_{i'} \;\middle|\; |S_{i'-1,r}| \ge 32k\epsilon^{-1}\ln n\right] \le \ell \cdot e^{-5k\ln n} = o(n^{-2k}).$$

A similar argument shows that when the procedure halts we also have $\Pr\left[|T_{j_r,r}| \le n^{2/3}\right] = o(n^{-2k})$.

Now, to finish the short path from s to t, we generate the random arcs of R between S_{i_r} and T_{j_r}

$$\Pr\left[R \cap (S_{i_r} \times T_{j_r}) = \emptyset \;\middle|\; |S_{i_r,r}| \ge n^{2/3} \wedge |T_{j_r,r}| \ge n^{2/3}\right] \le (1-p)^{n^{4/3}} \le e^{-\epsilon n^{1/3}} = o(n^{-2k}).$$

Putting all the pieces together, we have k disjoint paths, each consisting of a path of length at most $32\epsilon^{-1}\ln n$, followed by at most 2ℓ paths of length $d+1$ from $\bar{D}$ joined by edges from R, and finishing with a path of length at most $32\epsilon^{-1}\ln n$, for total length which numerical calculation shows is less than $100k\epsilon^{-1}\ln n$.

Since there are less than n^{2k} choices for the terminal pairs, the union bound shows that all choices of $2k$ nodes have short vertex disjoint paths linking them **whp**.

To conclude, we apply Lemma 6, which shows that these short paths will be discovered **whp** in polynomial time.

□

7 Conclusion

Spielman and Teng introduced smoothed analysis to help explain the success of the simplex algorithm. We have used smoothed analysis to examine the complexity of strong connectivity and

(s,t)-connectivity. In the analysis of **NP**-hard optimization problems, one can judge the degree of difficulty based on approximability. Here we provide another measure of difficulty, based on the existence of heuristics for smoothed instances. We find that, according to this measure, strong connectivity seems easier than (s,t)-connectivity. This claim is somewhat surprising, since we determine if a graph is strongly connected by repeatedly checking if pairs of vertices are (s,t)-connected. However, strong connectivity is a more global property, so much so that in an instance like that of Section 5, even though there exists *some* pair which Algorithm $\mathcal{A}$ incorrectly concludes is not connected, there will *almost always* be another pair which Algorithm $\mathcal{A}$ *correctly* concludes is not connected, so the net result will be correct.

There are several directions for future research. First, it is easy to come up with a measure of difficulty; it is not easy to come up with a good measure. This paper represents a piece of "experimental data". Are there other problems which appear solvable or insolvable by heuristics on smoothed instances? Does this property seem to relate to the difficulty of these problems in practice? This is especially interesting in the case of **NP**-complete problems. (Questions of this nature are investigated by Beier and Vöcking in [5].)

Second, it would be nice if Theorem 4 was about **BPL** instead of almost-**L**. It is not clear how to achieve this, however. This complication seems related to the limitations of log-space computation. An analogous result holds (by the same proof, even) for the possibility of heuristics for recognizing if a digraph contains edge disjoint paths connecting 2 terminal pairs. In that case, the worst-case problem is **NP**-complete, and since the reduction has room to store the perturbed copy, we can show that if $\mathbf{NP} \not\subseteq \mathbf{BPP}$ then no heuristic is successful on smoothed instances. A similar question addresses the growing out-degree case, as in Section 3. We know Algorithm $\mathcal{A}$ does not work, but does some other heuristic? It would be nice to have a result of a form similar to Theorem 4 suggesting no heuristic works on graphs with unbounded out-degree.

Finally, it would be natural to extend these results to computing k-strong-connectivity and being k-linked to work for smoothed instances. Here we face some unresolved technical difficulties.

References

[1] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff, Random walks, traversal sequences, and the complexity of maze problems, *Proc. 20th IEEE Symp. on the Foundations of Computer Science* (1979), 218-223.

[2] N. Alon, A. Gyárfás, and M. Ruszinkó, Decreasing the diameter of bounded degree graphs, *J. Graph Theory* 35 (2000), 161-172.

[3] N. Alon and N. Kahale, A spectral technique for coloring random 3-colorable graphs, *DIMACS TR-94-35*, 1994.

[4] C. Banderier, K. Mehlhorn, and R. Beier. Smoothed analysis of three combinatorial problems, *Proc. 28th Intl. Symp. on the Mathematical Foundations of Computer Science* (2003).

[5] R. Beier and B. Vöcking, Typical properties of winners and losers in discrete optimization, Proc. of the 36th annual ACM Symposium on Theory of Computing, New York, ACM (2004) 343–352.

[6] B. Bollobás, Random Graphs, *Second Edition, Cambridge University Press* (2001).

[7] B. Bollobás and F. Chung, The Diameter of a Cycle Plus a Random Matching, *SIAM Journal on Discrete Mathematics* 1(3) (1988) 328-333.

[8] L. Bechetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schaefer, and T. Vredeveld, Smoothening Helps: A probabilistic analysis of the multi-level feedback algorithm, *FOCS03* (2003).

[9] A. Blum and J. Spencer, Coloring Random and Semi-Random k-Colorable Graphs, *Journal of Algorithms* 19 (1995) 204-234.

[10] T. Bohman, A.M. Frieze and R. Martin, How many random edges make a dense graph Hamiltonian? *Random Structures and Algorithms* 22 (2003) 33-42.

[11] T. Bohman, A.M. Frieze, M. Krivelevich and R. Martin, Adding random edges to dense graphs, Random Structures and Algorithms 24 (2004) 105-117.

[12] F. Chung and M. Garey, Diameter bounds for altered graphs, *Journal of Graph Theory* 8 (1984) 511-534.

[13] C. Cooper and A. M. Frieze, The size of the largest strongly connected component of a random digraph with a given degree sequence *Combinatorics, Probability and Computing* 13 (2004) 319-338.

[14] M. Faloutsos, P. Faloutsos and C. Faloutsos, On powerlaw relationships of the Internet topology, *SIGCOMM 99* (1999) 251-262.

[15] U. Feige and R. Krauthgamer, Finding and certifying a large hidden clique in a semirandom graph, *Random Structures and Algorithms*, 16:2 (2000) 195-208.

[16] U. Feige and J. Kilian, Heuristics for Semirandom Graph Problems, *Journal of Computer and System Sciences* 63 (2001) 639-671.

[17] U. Feige, A Tight Upper Bound on the Cover Time for Random Walks on Graphs, *Random Structures and Algorithms* 6 (1995) 51-54.

[18] O. Goldreich, S. Goldwasser, D. Ron, Property testing and its connection to learning and approximation, *J. ACM* 45 (1998) 653-750.

[19] O. Goldreich, D. Ron, Property testing in bounded degree graphs, *Proc. 29th STOC* (1997) 406-415.

[20] S. Janson, T. Łuczak and A. Ruciński, *Random graphs*, Wiley-Interscience (2000).

[21] N. D. Jones, Space-bounded reducibility among combinatorial problems, *Journal of Computer and System Sciences* 11 (1975) 68-75.

[22] R. Karp, The transitive closure of a random digraph, *Random Structures and Algorithms* 1 (1990) 73-94.

[23] M. Krivelevich, B. Sudakov, and P. Tetali, On smoothed analysis in dense graphs and formulas, *Random Structures and Algorithms* 29:2 (2006) 180–193.

[24] N. Nisan, On read-once vs. multiple access randomness in logspace, *Theoretical Computer Science* 107:1 (1993) 135-144.

[25] O. Reingold, Undirected ST-connectivity in log-space, *Proc. of the 37th annual ACM Symposium on Theory of Computing* (2005) 376-385.

[26] M. Santha and U. Vazirani, Generating quasi-random sequences from semi-random sources, *Journal of Computer and System Sciences* 33 (1986) 75-87.

[27] M. Sipser, *Introduction to the Theory of Computation*, PWS (1996).

[28] D. Spielman and S.H. Teng, Smoothed Analysis of Algorithms: Why The Simplex Algorithm Usually Takes Polynomial Time, *Proc. of the The Thirty-Third Annual ACM Symposium on Theory of Computing*, (2001) 296-305.

[29] D. Spielman and S.H. Teng, Smoothed Analysis: Motivation and Discrete Models, *Proc. of WADS 2003, Lecture Notes in Computer Science*, Springer-Verlag (2003).

[30] C. R. Subramanian, M. Fürer, and C. E. Veni Madhavan, Algorithms for coloring semi-random graphs, *Random Structures and Algorithms* 13:2 (1998) 125-158.

[31] H. Vollmer and K.W. Wagner, Measure one results in computational complexity theory, *Advances in Algorithms, Languages, and Complexity* (1997) 285-312.

A NL-completeness

The standard proof of the **NL**-completeness of (s,t)-connectivity makes nodes correspond to machine configurations, and includes out-edges for each pair of machine configurations which can follow directly one after the other. Since Turing machines have a finite set of symbols, there is a finite set of configurations which can follow a given configuration. So the reduction produces a graph with bounded in-degree and out-degree, and (s,t)-connectivity of bounded out-degree graphs is **NL**-complete.

Now, given a bounded degree instance of (s,t)-connectivity, we reduce it to an instance of strong connectivity by, for each node $i \neq s,t$, adding an arc from i to s and an arc from t to i. This does not add any path from s to t, so not-connected instances stay not-connected. If the original instance contained an (s,t)-path, the new instance is strongly connected, since there is an arc from any vertex to s, a path from s to t, and an arc from t to any vertex.

Unfortunately, this does not have bounded degree, since we increased the out-degree of t and in-degree of s both to $n-2$. To avoid this, we add 2 complete undirected binary trees with depth $\lceil \log(n-2) \rceil$, (realized by directed arcs appearing in both directions for each undrected edge). Then we direct an arc from t to the root of tree one, and also add arcs from leaves to the vertices of the original graph (with at most one arc added per leaf). This has the same effect as adding the arcs directly from t to everything, but increases the out-degree of t by 1 and adds $\mathcal{O}(n)$ vertices with in-degree and and out-degree at most 3. Similarly, we connect the root of tree two to s and connect each vertex of the original graph to a leaf of tree two, at most one vertex per leaf. This has the same effect as adding the arcs directly from everything to s, but increases the in-degree of

s by 1 and adds $\mathcal{O}(n)$ vertices with in-degree out-degree at most 3. This transformation can be implemented by a log-space transducer, since it only requires a little bit-shifting to produce the binary tree. □

procedure `GenerateSets2`(Disjoint paths $P_1, P_2, \ldots, P_k$)

$U := V$

for $r = 1, \ldots, r^\star$ **do**
  $S_{0,r} :=$ first $32\epsilon^{-1} \ln n$ nodes of P_r.
  $T_{0,r} :=$ last $32\epsilon^{-1} \ln n$ nodes of P_r.
  $U := U \setminus N_d(S_{0,r} \cup T_{0,r})$
  $i_r := 0$
  $j_r := 0$
end for
for $r = r^\star + 1, \ldots, k$ **do**
  $U := U \setminus \{\text{nodes of } P_r\}$
end for

while $\exists r$: $(|S_{i,r}| \leq n^{2/3}$ and $i_r \leq \ell)$ or $(|T_{j,r}| \leq n^{2/3}$ and $j_r \leq \ell)$ **do**
  if $|S_{i_r}| \leq n^{2/3}$ and $i_r \leq \ell$ **then**
    $S_{i+1,r} := \emptyset$
    for all $s' \in S_{i,r}$ **do**
      if $|S_{i+1,r}| \leq n^{2/3}$ and there exists $s'' \in U$ such that $(s', s'') \in R$ **then**
        $S_{i+1,r} := S_{i+1,r} \cup N_d^+(\{s''\})$
        $U := U \setminus N_d(S_{i+1,r})$
      end if
    end for
    $i_r := i_r + 1$
  end if
  if $|T_{j,r}| \leq n^{2/3}$ and $j_r \leq \ell$ **then**
    $T_{j+1,r} := \emptyset$
    for all $t' \in T_{j,r}$ **do**
      if $|T_{j+1,r}| \leq n^{2/3}$ and there exists $t'' \in U$ such that $(t'', t') \in R$ **then**
        $T_{j+1,r} := T_{j+1,r} \cup N_d^-(\{t''\})$
        $U := U \setminus N_d(T_{j+1,r})$
      end if
    end for
    $j_r := j_r + 1$
  end if
end while

Figure 2: Pseudocode to generate $S_{i,r}$ and $T_{j,r}$ for $r = 1, \ldots, r^\star$