Parameterized Streaming Algorithms for Topological Sorting

Ho-Lin Chen
National Taiwan University, Taipei, Taiwan

Peng-Ting Lin
Academia Sinica, Taipei, Taiwan

Meng-Tsung Tsai
Academia Sinica, Taipei, Taiwan

Abstract

Computing a topological ordering for an n-node directed acyclic graph (DAG) G is computationally challenging in streaming models. Chakrabarti et al. [SODA 2020] showed that in the insertion-only streaming model, every single-pass algorithm requires $\Omega(n^2)$ space, and every k-pass algorithm requires $n^{1+\Omega(1/k)}$ space for any constant $k \geq 1$.

We study the parameterized complexity of streaming algorithms for topological sorting, considering two parameters: the independence number α and the maximum displacement δ. Our results include an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm and an $O(n^{1/2})$-pass $O(n + \delta^2)$-space streaming algorithm. For dense random DAGs, both α and δ are small, allowing us to improve the state-of-the-art for topological sorting in random DAGs.

As applications, we show that strongly connected components (SCC) decomposition and 2-satisfiability (2-SAT) can be solved in $O(1/\varepsilon)$ passes using $O(\alpha n^{1+\varepsilon})$ space and $O(\alpha_I n^{1+\varepsilon})$ space, respectively, where α_I denotes the independence number of the implication graph induced by the input 2-SAT instance.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Independence Number, Chain Cover, SCC Decomposition, 2-Satisfiability

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.18

Funding This research was supported in part by the National Science and Technology Council under contract NSTC 114-2221-E-001-023 and 113-2221-E-002-204-MY3.

Acknowledgements We want to thank the anonymous reviewers for their valuable comments.

1 Introduction

We study the problem of computing a topological ordering of a given n-node directed acyclic graph (DAG) $G = (V, E)$ using space subquadratic in the number of nodes. Specifically, the goal is to output an ordering $(v_1, v_2, \ldots, v_n)$ of the nodes in V using $O(n^{2-\varepsilon})$ space for some constant $\varepsilon > 0$, such that for every $i, j \in [n]$[1] with $j > i$, there is no directed path in G from v_j to v_i. This problem has been studied by Chakrabarti et al. [5], who showed that in the insertion-only streaming model, every single-pass algorithm requires $\Omega(n^2)$ space, and every k-pass algorithm requires $n^{1+\Omega(1/k)}$ space for any constant $k \geq 1$.

For problems that require substantial space when computed with few passes in the streaming model, a natural direction is to study their parameterized complexity. This approach was introduced by Fafianie and Kratsch for edge dominating set [10] and by

[1] We define $[n] \coloneqq \{1, 2, \ldots, n\}$.

Chitnis et al. for maximal matching and vertex cover [7]. Since then, the parameterized complexity of various problems has been explored. Chitnis and Cormode studied treewidth and dominating set [6]. Agrawal et al. studied Min-Ones SAT [1]. Oostveen and van Leeuwen studied diameter and connectivity [21]. Lokshtanov et al. studied feedback vertex set and multiway cut [17].

Our model of computation follows the *graph streaming model* [20, 18]. The edges of the input graph $G = (V, E)$ are presented in an order determined by an adversary who has access to the algorithm but not its random seed. The algorithm processes the edges sequentially, making up to p passes over the input while using at most s space. During each pass, the algorithm can only read the stream forward and cannot backtrack. We refer to such an algorithm as a p-pass s-space streaming algorithm. If the edges are static, meaning only edge insertions occur in the stream, the model is called the *insertion-only model*. In contrast, if both edge insertions and deletions are allowed, where deletions can only remove previously inserted edges, the model is referred to as the *turnstile model*.

Our first main result shows that any n-node directed acyclic graph G with independence number α can be topologically sorted in $O(\log n)$ passes using $O(\alpha n)$ space. A directed graph G has *independence number* α if there exists a set of α nodes in G such that no two nodes in the set are joined by an edge in G, and no larger set satisfies this property. The number of passes can be further reduced to $O(1/\varepsilon)$ for every $\varepsilon > 0$ if $O(\alpha n^{1+\varepsilon})$ space is available. Additionally, we provide a smooth tradeoff between passes and space. Formally, we have:

▶ **Theorem 1.** *Let G be an n-node directed acyclic graph with independence number α. For every $\varepsilon > 0$, there exists an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space deterministic streaming algorithm to topologically sort G in the insertion-only model. Moreover, there is a smooth tradeoff between passes and space: for every constant $r \in (0, 1)$, there exists an $O(n^r)$-pass $O(n + \alpha n^{1-r/2} \log n)$-space streaming algorithm with the same functionality with probability $1 - 1/n^{\Omega(1)}$.*

Another main result is that any n-node directed acyclic graph G with an advice A of maximum displacement δ can be topologically sorted in $O(n^{1/2})$ passes using $O(n + \delta^2)$ space. For an n-node acyclic graph $G = (V, E)$, a function $A : V \to \mathbb{Z}$ is an *advice* of G with *maximum displacement* δ if G admits a topological ordering $(v_1, v_2, \ldots, v_n)$ such that for every $i \in [n]$, the displacement satisfies $|i - A(v_i)| \leq \delta$. Note that the function A is not required to be injective. We provide also a smooth tradeoff between passes and space. Formally, we have:

▶ **Theorem 2.** *Let G be an n-node directed acyclic graph with an advice A of maximum displacement δ. For every $t \in [n]$, there exists an $O(n/t)$-pass $O(n+t\delta+\delta^2)$-space deterministic streaming algorithm in the insertion-only model to topologically sort G.*

The above two algorithms can be applied to random directed acyclic graphs (random DAGs) $G \sim \mathcal{G}(n, p)$, where each edge is included independently with probability p. In random DAGs, the independence number can be estimated quite accurately, and an advice A can be provided using the in-degree of the nodes. Using these facts, we have an $\tilde{O}(1)$-pass $\tilde{O}(np^{-1})$-space streaming algorithm and an $O(\sqrt{np})$-pass $\tilde{O}(n)$-space streaming algorithm. These algorithms have almost identical pass and space requirement as [5]. But, unlike [5] which requires a chain that goes through all nodes, our algorithms apply to random DAGs $\mathcal{G}(n, p)$.

Furthermore, both of our algorithm exhibit smooth trade-offs between pass and space. Also, both of our algorithms and their analysis directly applies to graphs of type $G \cup H$, where $G \sim \mathcal{G}(n, p)$ is a random DAG and H can be any given graph with maximum in-degree $o(\sqrt{np \log n})$ that has the same node set and the same topological ordering as G.

The final main result gives a deterministic algorithm in the turnstile model to compute a topological ordering for any directed acyclic graph, stated below:

▶ **Theorem 3.** *Let G be an n-node directed acyclic graph. For every $k \in [n]$, there exists an k-pass $O(n^2/k)$-space deterministic streaming algorithm in the turnstile model to topologically sort G.*

1.1 Technical Ingredients

If a directed acyclic graph $G = (V, E)$ has independence number α, then every node-induced subgraph of G admits a chain cover of at most α chains. This is a stronger condition than simply assuming that a largest antichain in G contains at most α nodes, a parameter studied extensively for algorithms on a RAM [16, 4], as our condition ensures a structural property that holds for all node-induced subgraphs, enabling a divide-and-conquer approach to compute a chain cover consisting of few chains. Since the chain cover consists of α chains, the set of nodes reachable from any node x in G can be succinctly represented by a tuple of α integers. Combined with a distributed variant of the Bellman-Ford algorithm, this forms the basis of our first algorithm.

Our second and third algorithms rely on the concept of a *source subgraph*, a node-induced subgraph of G defined by a set S such that no edge in G enters S from $V \setminus S$. The second algorithm leverages the maximum displacement δ to construct a subgraph H that fits in memory and contains an r-node source subgraph of G for some sufficiently large r. The third algorithm refines G by removing a majority of out-neighbors from high-degree nodes, producing a sparse subgraph (in terms of edge density) that also contains a large source subgraph of G. Consequently, topological sorting of G can be performed by iteratively peeling away a source subgraph. Since each source subgraph contains a moderate number of nodes, the number of peeling rounds remains low.

1.2 Applications

Finally, in Section 7, we show that the idea used in our topological sort algorithm can be applied to SCC decomposition, yielding an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space deterministic algorithm for any $\varepsilon > 0$, as well as a ranodmized $O(k)$-pass $O((n^2 \log n)/k)$-space. Since SCC decomposition can be directly applied to solve 2-SAT, these results also extend to the 2-SAT problem.

1.3 Paper Organization

The paper is organized as follows. In Section 2, we introduce our notation. In Section 3, we develop algorithms for topologically sorting directed acyclic graphs with independence number α, proving Theorem 1. In Section 4, we design algorithms for topologically sorting directed acyclic graphs given an advice function with maximum displacement δ, thereby proving Theorem 2. We then combine the results from Section 3 and Section 4 to obtain a result for random DAGs in Section 5. In Section 6, we introduce a deterministic algorithm in the turnstile model for topologically sorting general DAGs. Finally, in Section 7, we discuss selected applications of topological sorting, including SCC decomposition and 2-satisfiability.

2 Notation

Our input graph is an n-node directed acyclic graph $G = (V, E)$ in which each directed edge $(u, v) \in E$ is from node u to node v. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and $[a, b]$ (resp. (a, b)) denote the closed (resp. open) interval between a and b. A *topological ordering* of G is an ordering $(v_1, v_2, \ldots, v_n)$ of the nodes in V such that there is no directed path in G from v_j to v_i for any $j > i$ with $i, j \in [n]$. We use $G[S]$ to denote the *node-induced subgraph* of G induced by the node set S; that is, $G[S] = (S, E^S)$ where all edges in E with both endnodes in S form E^S. We use $G \sim \mathcal{G}(n, p)$ to denote an n-node random directed acyclic graph G where each edge is included in G with probability p independently, and orient the directions of all edges based on a random permutation on the n nodes from an earlier node in the permutation to a later node.

For each node x in G, if there is a directed edge $(x, y) \in E$, we say that y is an *out-neighbor* of x, and x is an *in-neighbor* of y. A node x with no in-neighbors (resp. no out-neighbors) is called a *source* (resp. a *sink*). We say that a node x has *in-degree* (resp. *out-degree*) k if it has k in-neighbors (resp. out-neighbors), denoted as $d_{in}(x) = k$ (resp. $d_{out}(x) = k$).

The *transitive closure* of a directed graph $G = (V, E)$ is the graph $G^{tc} = (V, E^{tc})$, where $(u, v) \in E^{tc}$ if and only if there exists a directed path from u to v in G. A *chain* of G is a sequence of nodes $(v_1, v_2, \ldots, v_t)$ for some integer $t \geq 1$, such that $(v_1, v_2, \ldots, v_t)$ forms a directed path in the transitive closure G^{tc}. A *chain cover* of G is a collection of chains such that the nodes contained in the chains partition the node set V. An *antichain* of G is a set of nodes such that no directed path exists between any two distinct nodes u, v in the set.

We say that a directed graph G has *independence number* α if there exists a set of α nodes in G such that no two nodes in the set are joined by an edge in G, and no larger set satisfies this property.

For an n-node acyclic graph $G = (V, E)$, we say that a function $A : V \to \mathbb{Z}$ is an *advice* of G with *maximum displacement* δ if G admits a topological ordering $(v_1, v_2, \ldots, v_n)$ such that for every $i \in [n]$, the displacement satisfies $|i - A(v_i)| \leq \delta$.

In this paper, the space usage of each algorithm is measured in words, while the space lower bound of each problem is measured in bits. Therefore, if an algorithm's space usage matches the problem's space lower bound, the algorithm is considered nearly optimal in space usage, up to an $O(\log n)$ factor.

3 An $O(1/\varepsilon)$-Pass $O(\alpha n^{1+\varepsilon})$-Space Algorithm and Tradeoff

We begin with showing that the transitive closure of an n-node directed acyclic graph G with independence number α can be represented by the transitive closure of an $O(\alpha n)$-edge subgraph H of G. Since G and H have the same transitive closure, computing a topological ordering of H suffices to topologically sort G. Then, we devise a streaming algorithm in the insertion-only model to compute H, which yields an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm to topologically sort any given n-node acyclic directed graph with independence number α. Moreover, there is a smooth tradeoff between passes and space: for every constant $r \in (0, 1)$, there exists an $O(n^r)$-pass $O(n + \alpha n^{1-r/2} \log n)$-space streaming algorithm with the same functionality with probability $1 - 1/n^{\Omega(1)}$. This proves Theorem 1.

▶ **Lemma 4.** *Every directed acyclic graph G with independence number α contains an $O(\alpha n)$-edge subgraph H such that G and H have the same transitive closure. Moreover, if a graph G' has the same transitive closure as G, then G' also contains an $O(\alpha n)$-edge subgraph with the same transitive closure, even if the independence number of G' exceeds α.*

Proof. Let S be an antichain in G; that is, a subset of nodes in G such that for any two nodes u and v in the subset, there exists no directed path from u to v. Thus, S is an independent set and $|S| \leq \alpha$. This property holds also for any largest antichain in G. By Dilworth's theorem [9], G has a chain cover consisting of at most α chains. For each node x in G, if x has out-degree greater than α, then by the pigeonhole principle, there exist two out-neighbors u and v of x that belong to the same chain in the chain cover. Let the chain be $a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_t$ for some $t \geq 2$ and $u = a_i, v = a_j$ for some $i < j$. If the edge (x, v) is removed from G, then the resulting graph has the same transitive closure as G. This holds because x can still reach v via the edge (x, u) followed by a directed path P that witnesses the chain $a_i \rightarrow a_j$. Note that x is not on P, as G is acyclic, and neither is the edge (x, v). Consequently, one can iteratively remove an edge from G while retaining the transitive closure unchanged until every node in G has out-degree at most α.

The same argument works for any graph G' that has the same transitive closure as G, even if the independence number of G' exceeds α. ◀

One may find that Lemma 4 still holds if the number of nodes in a largest antichain in G is α. However, it is essential for our algorithm to have the independence number of G bounded by α. In this way, every node-induced subgraph of G (rather than only G) has an antichain consisting of at most α chains, which is the key to find the subgraph H efficiently in the streaming model.

First, we find the chain cover used in the proof of Lemma 4 as follows.

▶ **Lemma 5.** *Let G be an n-node acyclic directed graph with independence number α. For every $\varepsilon > 0$, there exists an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm that outputs a chain cover of G consisting of α chains.*

Proof. We begin with an $O(\log n)$-pass $O(\alpha n)$-space streaming algorithm. First, we partition the nodes of G into subsets $\{S_{2i-1}, S_{2i} : i \in [t]\}$ arbitrarily for some integer $t = O(n)$, each containing $O(1)$ nodes. We then scan the stream of all edges in G once. For each $j \in [2t]$, we collect all edges in the node-induced subgraph $G[S_j]$ in memory. Next, we apply the approach from the proof of Lemma 4 to compute a chain cover consisting of α chains for $G[S_j]$ and trim $G[S_j]$ into a sparse subgraph with $O(\alpha|S_j|)$ edges.

We then scan the stream of all edges in G again. For each pair of consecutive node-induced subgraphs $G[S_{2i-1}]$ and $G[S_{2i}]$, we collect edges with one endnode in S_{2i-1} and the other in S_{2i} (i.e., crossing edges). Since we have a chain cover consisting of α chains for both $G[S_{2i-1}]$ and $G[S_{2i}]$, at most α essential crossing edges per node in S_{2i-1} and S_{2i} are required to preserve the transitive closure. This can be done using $O(\alpha(|S_{2i-1}| + |S_{2i}|))$ space because, for each of the α chains in the node-induced subgraph S_{2i-1} (resp. S_{2i}), each node x in S_{2i} (resp. S_{2i-1}) only needs to keep the single out-neighbor y that appears first in the chain. Finally, we replace the union of $G[S_{2i-1}]$, $G[S_{2i}]$, and their essential crossing edges with an $O(\alpha(|S_{2i-1}| + |S_{2i}|))$-edge subgraph H, ensuring that H and $G[S_{2i-1} \cup S_{2i}]$ have the same transitive closure and thus the chain cover of H consists of at most α chains. Such a sparse subgraph H can be found because the union of $G[S_{2i-1}]$, $G[S_{2i}]$, and the essential crossing edges has the same transitive closure as $G[S_{2i-1} \cup S_{2i}]$ and hence we can apply Lemma 4 though the independence number of the union may exceed α.

The above process reduces the number of subsets by half. The space usage remains $O(\alpha n)$. Repeating this process for $O(\log n)$ iterations merges all subsets into a single subgraph with a chain cover of α chains.

The above approach groups two subsets at a time. More generally, we can group n^ε subsets at a time. This increases space usage by a factor of n^ε while reducing the number of passes to

$$O(\log_{n^\varepsilon} n) = O(1/\varepsilon). \qquad \blacktriangleleft$$

Then, we use the chain cover found in Lemma 5 to compute H as follows.

▶ **Corollary 6.** *Let G be an n-node acyclic directed graph with independence number α. For every $\varepsilon > 0$, there exists an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm that outputs an $O(\alpha n)$-edge subgraph H such that G and H have the same transitive closure.*

Proof. By Lemma 5, we obtain a chain cover of G consisting of α chains using the claimed number of passes and space. Then, in another pass over all edges of G, for each node x, we store all of its out-neighbors in a list. If the list exceeds α out-neighbors, at least one can be discarded, as established in the proof of Lemma 4. ◀

Finally, given the subgraph H, a topological ordering of G can be computed by topologically sorting H. This gives an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space deterministic streaming algorithm to compute a topological ordering for any n-node directed acyclic graph G in the insertion-only model. This proves the few-pass algorithm stated in Theorem 1.

3.1 Tradeoff Between Passes and Space

We give a smooth tradeoff between passes and space: for every constant $r \in (0, 1)$, there exists an $O(n^r)$-pass $O(n + \alpha n^{1-r/2})$-space streaming algorithm with the same functionality. Our divide-and-conquer approach is inspired by the parallel topological sorting method of Schudy [22]. However, since reachability is computationally expensive in streaming models, we overcome this limitation by leveraging the concept of chain cover, which is a novel aspect of our approach compared to Schudy's result.

▶ **Lemma 7.** *Given an n-node directed acyclic graph G and a topological ordering of $\Omega(k \log n)$ nodes, sampled independently and uniformly at random (possibly with repetition), there exists an $O(n/k)$-pass $O(n)$-space streaming algorithm that topologically sorts G with probability $1 - 1/n^{\Omega(1)}$.*

Proof. We add a new node s to G, with a directed edge to every node in G that has in-degree 0. Let R be a set of $\Omega(k \log n)$ sampled nodes, denoted as $R = \{v_1, v_2, \ldots, v_{|R|}\}$, such that no directed path in G starts from v_i to v_j for any $i > j$, where $i, j \in [|R|]$. Define $v_0 := s$.

For each integer $i \in [0, |R|]$, let U_i be the set of nodes in G that are reachable from v_i but not from any v_j with $j > i$. Since s can reach every node in G, the sets U_i for all integer $i \in [0, |R|]$ form a partition of V. Therefore, storing all U_i in memory needs $O(n)$ space. Given this partition, a topological ordering of G can be obtained by first sorting the nodes within each U_i independently and then concatenating their orderings.

Since R is a randomly selected subset of $\Omega(k \log n)$ nodes, the sets U_i satisfy the following property:

▷ Claim 8. For each integer $i \in [0, |R|]$ and each node $x \in U_i$, any longest path in G (with unweighted edges) from v_i to x contains at most n/k edges with probability $1 - 1/n^{\Omega(1)}$.

Proof. Let V be the set of nodes in G, including s. Consider any pair of nodes $y, z \in V$ such that z is reachable from y. Fix a longest path P_{yz} from y to z, and let $\mathcal{C}$ be the collection of such paths P_{yz} whose lengths exceed n/k. Since there are at most $\binom{n}{2}$ such pairs, we have $|\mathcal{C}| \leq \binom{n}{2}$.

The probability that none of the internal nodes in P_{yz} (excluding y and z) is included in R is

$$(1 - 1/k)^{\Omega(k \log n)} = 1/n^{\Omega(1)}.$$

Applying the union bound, we conclude that with probability at least $1 - 1/n^{\Omega(1)}$, every path in $\mathcal{C}$ contains at least one internal node in R.

Consequently, if there exists a node $x \in U_i$ such that a longest path P from v_i to x contains more than n/k edges, then with probability at least $1 - 1/n^{\Omega(1)}$, some internal node v_j in P belongs to R. Since there is a path from v_i to v_j, we have $j > i$, contradicting the assumption that x is in U_i. ◁

To match the claimed number of passes and space, we design a distributed variant of the Bellman-Ford algorithm (stated also in Algorithm 1). Assign a weight of -1 to all edges in G. Then, the shortest distance from s to any node $x \in V$ corresponds to the length of the longest path (in terms of edge count) from s to x. Sorting the nodes by their distances from s yields a topological ordering. However, computing the distances from s may require many passes or substantial space.

Instead of running the Bellman-Ford algorithm from a single source node s, we execute it in parallel from every node in the set R. If a node x is reachable from two nodes v_i and v_j with $i < j$, we discard the information from v_i, since x does not belong to U_i. In Algorithm 1, this behavior is enforced by the rank function r, which is set such that $r(v_i) > r(v_j)$ for all $i, j \in [0, |R|]$ with $i < j$, ensuring that only information from the appropriate source is retained. As a result, the space usage $O(1)$ per node and $O(n)$ in total. By Claim 8, the Bellman-Ford algorithm terminates after $O(n/k)$ iterations.

The node ordering returned by Algorithm 1 is a topological ordering of G because it topologically sorts the nodes in each U_i by their longest distances (in term of edge count) from v_i. Moreover, the concatenation of the topological orderings of nodes in U_is for all $i \in [0, |R|]$ cannot have any edge (x, y) in G for some $x \in U_j$ and $y \in U_i$ with $i < j$, as it would violate the definition of the sets U_i. ◀

▶ **Lemma 9.** *Given an n-node directed acyclic graph G, a chain cover of G consisting of β chains, and $\Omega(k \log n)$ nodes sampled uniformly at random from G independently (possibly with repetition), there exists an $O(n/k)$-pass $O(n + \beta k \log n)$-space streaming algorithm that topologically sorts these sampled nodes with probability $1 - 1/n^{\Omega(1)}$.*

Proof. We topologically sort the sampled nodes by ordering them based on the cardinalities of their reachable sets. Let R be the set of sampled nodes. For each $x \in R$, we perform a depth-$O(n/k)$ BFS rooted at x. The β chains can be stored in memory using $O(n)$ space. Each BFS computation requires an additional $O(\beta)$ space, as detailed below, and across all BFSs, the exploration of nodes at depth d is performed simultaneously. Consequently, the total number of passes is $O(n/k)$.

We execute each of these BFSs as follows. The key difference between our BFS and a standard BFS is that we define the *child nodes* of any node x to include both its out-neighbors and the out-neighbors of all descendant nodes of x along the chain to which x belongs. Since we are only interested in the set of nodes that x can reach, we do not need to maintain the

Algorithm 1 A distributed Bellman-Ford algorithm for topological sorting.

```
Input: A directed graph $G = (V \cup \{s\}, E)$ where $s$ can reach all nodes and all edges
       have weight $-1$, a random subset $R \subseteq V$, and a rank function
       $r : R \cup \{s\} \to \mathbb{N}$ such that no directed path in $G$ starts from a node with a
       smaller $r$-value to one with a larger $r$-value.
Output: A topological ordering of $G$.
$d[x] \leftarrow (\infty, \infty)$ for all $x \in V$;
$d[y] \leftarrow (r(y), 0)$ for all $y \in R \cup \{s\}$;
for $i = 1$ to $t = O(n \log n/|R|)$ do
    foreach $(u, v) \in E$ do
        if $d[u] + (0, -1) < d[v]$ then
            // coordinate-wise additions and comparisons, prioritizing the first
               coordinate
            $d[v] \leftarrow d[u] + (0, -1)$;
        end
    end
end
return nodes $x \in V$ in non-decreasing order of $d$ values;
```

full structure of a standard BFS. Thus, for each BFS level, we only need to store at most β nodes. This is because if two nodes belong to the same chain, where one is an ancestor of the other, then the child nodes of the ancestor form a superset of those of the descendant.

We only need to perform depth-$O(n/k)$ BFSs for all $x \in R$ to compute the reachable sets for all $x \in R$. For every pair of nodes $x, y \in G$, consider a shortest path from x to y. Let $\mathcal{C}$ be the collection of all such shortest paths of length at least n/k. For each path in $\mathcal{C}$, the probability that none of its internal nodes belong to R is

$$(1 - 1/k)^{\Omega(k \log n)} \leq 1/n^{\Omega(1)}.$$

Applying the union bound over all paths in $\mathcal{C}$, where $|\mathcal{C}| \leq \binom{n}{2}$, we conclude that with probability $1 - 1/n^{\Omega(1)}$, every path in $\mathcal{C}$ contains at least one internal node in R. Consequently, if there exists a node y that is reachable from x via a shortest path longer than n/k, then it must be visible from the $O(n/k)$-depth BFS rooted at some node in R. ◀

We are now ready to prove the tradeoff stated in Theorem 1. Let r be any constant in $(0, 1)$. First, we partition the node set of G into $n^{r/2}$ subsets arbitrarily, each containing $O(n^{1-r/2})$ nodes. For each subset S, we apply Lemma 5, setting $\varepsilon \to 1/\log n$ and $n \to n^{1-r/2}$, to obtain a chain cover of the subgraph induced by S, which consists of α chains. This holds because any node-induced subgraph of G also has independence number α. Combining these chain covers across all subsets yields a chain cover of G with $O(\alpha n^{r/2})$ chains.

Next, we use this chain cover as input for Lemma 9, setting $\beta = \alpha n^{r/2}$, and then apply Lemma 7 to compute a topological ordering of G, where the parameter k in both lemmas is set to n^{1-r}. Consequently, the number of passes is

$$O(n^{r/2} \log n + n/n^{1-r}) = O(n^r),$$

and the required space is

$$O(\alpha n^{1-r/2} + n + \alpha n^{r/2} n^{1-r} \log n) = O(n + \alpha n^{1-r/2} \log n).$$

This establishes the smooth tradeoff in Theorem 1.

3.2 Graph Classes with Small Independence Number

In what follows, we provide a brief proof (without claiming novelty) that a dense random DAG $G \sim \mathcal{G}(n,p)$ has a small independence number with high probability for every $p \in (0,1)$. Given the bounds on the number of nodes in a largest antichain in a random DAG due to Bollobás and Brightwell [3], our bound in Lemma 10 is nearly optimal, up to a logarithmic factor. We note that a result by Frieze [12] may eliminate this logarithmic factor, but it requires the assumption $p = o(1)$.

▶ **Lemma 10.** *For a random DAG $G \sim \mathcal{G}(n,p)$, where each edge is included independently with probability p, the independence number of G is $O(p^{-1}\log n)$ with probability $1 - 1/n^{\Omega(1)}$.*

Proof. Let $p = 1/k$, and define X_t as the number of independent sets of t nodes in G. Since $G \sim \mathcal{G}(n,p)$, we have

$$\mathbf{E}[X_{\lceil 3k\log n\rceil}] = \binom{n}{\lceil 3k\log n\rceil}(1-1/k)^{\binom{\lceil 3k\log n\rceil}{2}} \le n^{\lceil 3k\log n\rceil} e^{-1/k\binom{3k\log n}{2}} = e^{-\Omega(k\log^2 n)}.$$

Since X_t is non-negative, applying Markov's inequality gives

$$\Pr[X_{\lceil 3k\log n\rceil} \ge 1] \le e^{-\Omega(k\log^2 n)}.$$

Thus, with probability $1-1/n^{\Omega(1)}$, G contains no independent set of $\lceil 3k\log n\rceil = O(p^{-1}\log n)$ nodes or more. ◀

4 An $O(n^{1/2})$-Pass $O(n+\delta^2)$-Space Algorithm and Tradeoff

In this section, we present an $O(n/t)$-pass $O(n+t\delta+\delta^2)$-space topological sort algorithm assuming that an advice A with maximum displacement δ is given, where t can be any given number in $[n]$. This proves Theorem 2.

The main idea is that, given any $t \in [n]$, we devise a process that can find and sort a source subgraph of at least t nodes using a single pass and $O(t\delta+\delta^2)$ space. We say a node-induced subgraph $G[S]$ of G is a *source subgraph* if no edge in G is from a node outside S to a node in S. Our algorithm just repeats this process $O(n/t)$ times. In order to find a source subgraph, we need the following observation:

▶ **Lemma 11.** *Let G be an n-node directed acyclic graph with an advice A of maximum displacement δ. Given any $t \in [n]$. Let H_1 be the subgraph induced by all nodes v with $A(v) \le t+3\delta$ and H_2 be the subgraph induced by all nodes v with $A(v) \le t+\delta$. If H_3 is a subgraph of H_2 and a source subgraph of H_1, then H_3 is a source subgraph of G.*

Proof. It suffices to show that no edge goes from $G \setminus H_3$ to H_3. Since H_3 is a source subgraph of H_1, we only need to prove that that no edge goes from $G \setminus H_1$ to H_3.

Let σ be a topological ordering of G such that $|\sigma(v) - A(v)| \le \delta$ for all nodes v. For any node $u_1 \in H_3$, we have $\sigma(u_1) \le A(u_1)+\delta \le t+2\delta$. Similarly, for any node $u_2 \in G \setminus H_1$, we have $\sigma(u_2) > t+2\delta$. Combining the two inequalities, we have $\sigma(u_1) < \sigma(u_2)$. Therefore, no edge goes from $G \setminus H_1$ to H_3, which completes the proof. ◀

Using the above lemma, to find a source subgraph in G, it suffices to remember the induced subgraph H_1 and find the maximum source subgraph H_3 that satisfies the conditions listed. This can be done by remembering the entire induced subgraph H_1 in a single pass. The following two lemmas show that keeping the entire subgraph H_1 takes $O(t\delta+\delta^2)$ space, and the resulting source subgraph has size at least t.

▶ **Lemma 12.** *Let G be an n-node directed acyclic graph with an advice A of maximum displacement δ. Given any $t \in [n]$. There are at most $t + 4\delta$ nodes v with $A(v) \leq t + 3\delta$.*

Proof. Let σ be a topological ordering of G such that $|\sigma(v) - A(v)| \leq \delta$ for all nodes v. Only the first $t + 4\delta$ nodes of σ can satisfy $A(v) \leq t + 3\delta$. ◀

Lemma 12 shows that the subgraph H_1 defined in Lemma 11 has $O(t + \delta)$ nodes. Since the maximum displacement between the advice and the true ordering is at most δ, any pair of nodes whose advice values differ by more than 2δ must appear in the same relative order as suggested by the advice. Therefore, when computing a topological ordering, it suffices to retain only the edges between nodes whose advice values differ by less than 2δ. These edges will be referred to as *relevant* edges. The subgraph H_1 can be stored in $O(t\delta + \delta^2)$ space if we only keep the relevant edges.

▶ **Lemma 13.** *Let G be an n-node directed acyclic graph with an advice A of maximum displacement δ. Given any $t \in [n]$. Let H_1 be the subgraph induced by all nodes v with $A(v) \leq t + 3\delta$ and H_2 be the subgraph induced by all nodes v with $A(v) \leq t + \delta$. There exists a subgraph H_3 of size at least t such that H_3 is a subgraph of H_2 and a source subgraph of H_1.*

Proof. Let σ be a topological ordering of G such that $|\sigma(v) - A(v)| \leq \delta$ for all nodes v. The subgraph H induced by the first t nodes in σ has $A(v) \leq t + \delta$ and thus is a subgraph of H_2. Since H is the first t nodes in a topological ordering, H has no incoming edges and thus is a source subgraph of H_1. ◀

We are now ready to present the algorithm stated in Theorem 2. Given a graph G and an advice function A, the algorithm makes a single pass over the input and stores the subgraph H_1 induced by all nodes v with $A(v) \leq t + 3\delta$, along with all relevant edges between them. It then identifies the largest source subgraph of H_1 by iteratively expanding H_3, starting from the empty set and adding nodes whose advice values are at most $t + \delta$ and whose in-neighbors have all already been included in H_3. By Lemma 13, this source subgraph, denoted H_3, contains at least t nodes. The algorithm sorts the nodes of H_3 and places them at the front of the final topological ordering. Repeating this process $O(n/t)$ times yields a topological ordering of the entire graph G.

According to Lemma 12, the space requirement for keeping the subgraph H_1 in each step is $O(t\delta + \delta^2)$. $O(n)$ space is needed to keep the final topological ordering. Therefore, the total space usage for our streaming algorithm is $O(n + t\delta + \delta^2)$. Note that when $t = n$, the algorithm just remembers all relevant edges in a single pass using space $O(n\delta)$.

5 Random DAGs

In this section, we consider the topological ordering of random directed acyclic graphs (random DAGs) defined in Section 2. We apply our algorithms described in Section 3 and Section 4 to random DAGs in the streaming model and obtain a hybrid algorithm comprised of the two. We conclude this section by comparing our results to the results in [5].

First, for dense random DAGs, we can apply our algorithm in Section 3. Lemma 10 shows that, for a random DAG $G \sim \mathcal{G}(n, p)$, where each edge is included independently with probability p, the independence number of G is $O(p^{-1} \log n)$ with probability $1 - 1/n^{\Omega(1)}$. Substituting this number into our algorithm in Section 3, we have an $O(1/\varepsilon)$-pass $O(n^{1+\varepsilon} p^{-1} \log n)$-space streaming algorithm for every $\varepsilon > 0$. Choosing ε to be $(\log n)^{-1}$, we have an $\tilde{O}(1)$-pass $\tilde{O}(np^{-1})$-space streaming algorithm.

Second, we apply our topological sort algorithm described in Section 4. In order to apply the algorithm, we must first obtain an advice A. For a random DAG $G \sim \mathcal{G}(n,p)$, we can obtain an advice with maximum displacement at most $O(\sqrt{p^{-1} n \log n})$ with probability $1 - 1/n^{\Omega(1)}$ using the incoming degrees of the nodes. The same technique has been used in [5] while considering a variant of random directed acyclic graphs.

▶ **Lemma 14.** *Let $G = (V,E)$ be a random DAG. For every node $v \in V$, $|d_{in}(v)/p - \sigma(v)| = O(\sqrt{p^{-1} n \log n})$ with probability $1 - 1/n^{\Omega(1)}$.*

Proof. For each node $v \in V$, the in-degree $d_{in}(v)$ is a random variable that follows a binomial distribution with $\sigma(v) - 1$ trials and success probability p. By applying Chernoff Bound we have

$$\Pr\left[|d_{in}(v) - (\sigma(v)-1)p| = O(\sqrt{pn \log n})\right] = 1/n^{\Omega(1)}. \qquad ◀$$

By Lemma 14, $A(v) = d_{in}(v)/p$ is an advice with maximum displacement $O(\sqrt{p^{-1} n \log n})$ with high probability. The calculation of A can be done in one pass and $O(n)$ space by counting the incoming degrees of the nodes.

Using $A(v) = d_{in}(v)/p$ as an advice, Theorem 2 directly gives an $O(n/t)$-pass $O(n+t\delta+\delta^2)$-space streaming algorithm where $\delta = O(\sqrt{p^{-1} n \log n})$. Furthermore, notice that the space requirement $O(t\delta + \delta^2)$ is for keeping the $O(\delta)$ relevant edges for each of $O(t+\delta)$ nodes in H_1. However, in a random graph $\mathcal{G}(n,p)$, the probability that all subgraphs that could potentially be chosen as H_1 have only $O(pt\delta + p\delta^2)$ edges is at least $1 - 1/n^{\Omega(1)}$. As a result, the algorithm described in Theorem 2 only uses $O(n/t)$-pass $O(t\sqrt{pn \log n} + n \log n)$-space with probability $1 - 1/n^{\Omega(1)}$. Choosing t to be $\sqrt{np^{-1} \log n}$, we have an $O(\sqrt{np/\log n})$-pass $O(n \log n)$-space streaming algorithm.

Combining the above algorithms, we have two topological sort algorithms for random DAGs with almost identical (up to poly-logarithmic improvement) pass and space requirement as [5]. Unlike [5] which requires a chain that goes through all nodes, our algorithms apply to random DAGs $\mathcal{G}(n,p)$.

Furthermore, both of our algorithm exhibit trade-offs between pass and space. Also, both of our algorithms and their analysis directly apply to graphs of type $G \cup H$, where $G \sim \mathcal{G}(n,p)$ is a random DAG and H can be any given graph with maximum in-degree $o(\sqrt{np} \log n)$ that has the same node set and same node ordering as G. The independence number only decreases after adding the edge set of H to G, and the advice derived from Lemma 14 is still correct with high probability for sparse H.

6 A Deterministic Approach in the Turnstile Model

We devise a deterministic algorithm to compute a topological ordering for any directed acyclic graph in the turnstile model. This proves Theorem 3. Our technical ingredients for this deterministic result include a sparse certificate and the set reconciliation.

Let $G = (V,E)$ be an n-node directed acyclic graph. We say that a subgraph H of G is a *t-certificate* of topological sorting if for any topological ordering σ_H of H there exists a topological ordering σ_G of G so that the longest common prefix of σ_H and σ_G has length at least t. Our deterministic algorithm to compute σ_G (initialized as an empty sequence) is simply computing a topological ordering σ_H of a t-certificate H of G, appending the first t nodes in σ_H to σ_G, followed by reducing the problem to a subproblem. Note that the first t nodes in σ_H induces a source subgraph of G. See also Algorithm 2, whose correctness follows from the fact that the first t nodes in σ_H form a valid prefix of σ_G.

Algorithm 2 Deterministic topological sorting.

```
    input  : a directed acyclic graph $G = (V, E)$
    output : a topological ordering $\sigma_G$ of $G$
1   $\sigma_G \leftarrow$ null;
2   while $G$ is not an empty graph do
3       compute a $t$-certificate $H$ of $G$;
4       compute a topological ordering $\sigma_H$ of $H$;
5       append the first $t$ nodes in $\sigma_H$ to $\sigma_G$;
6       let $G$ be the subgraph of $G$ induced by the nodes not containing in $\sigma_G$;
7   end
8   return $\sigma_G$;
```

In what follows, we present how to obtain a sparse t-certificate H. For each node $v \in V$, define $N^-(v)$ to be the set of edges in E that connect v's in-neighbors to v. Define further that $N^-_\ell(v)$ to be any subset of $N^-(v)$ of cardinality exactly $\min\{\ell, |N^-(v)|\}$. We show in Lemma 15 that the union of $N^-_t(v)$ for all $v \in V$ is a t-certificate of $O(tn)$ edges.

▶ **Lemma 15.** *For every integer $t \geq 1$, the subgraph*

$$H = (V, F) \text{ where } F := \bigcup_{v \in V} N^-_t(v)$$

is a t-certificate of G consisting of $O(tn)$ edges.

Proof. Let σ_H be a topological ordering of H. Let S be the set of nodes placed at the first t positions of σ_H. Let $G[S]$ be the subgraph G induced by the node set S. It suffices to show that the concatenation of any topological ordering of $G[S]$ and $G[V \setminus S]$ is a topological ordering of G.

For any node $v \in S$, $N^-_t(v) < t$ because all in-neighbors of v in H have to precede v in σ_H. Combining this observation with the definition of $N^-_t(v)$, we have that for any node $v \in S$ $N^-_t(v) = N^-(v)$. Hence, no edges crossing $V \setminus S$ and S in G can be directed from $V \setminus S$ to S. Consequently, let σ_1 (resp. σ_2) be the topological ordering of $G[S]$ (resp. $G[V \setminus S]$), the concatenation $\sigma_1 \circ \sigma_2$ will not induce any backward edge. Because $G[S]$ and $G[V \setminus S]$ both are subgraphs of G and G is acyclic, this ensures the existence of σ_1 and σ_2. ◀

The remaining to show is how to obtain $N^-_t(v)$ for all v deterministically in the turnstile model. To achieve this, we use the characteristic polynomials introduced in the study of the set reconciliation problem [19]. The reason not to use ℓ_0-samplers [15] to obtain $N^-_t(v)$, a standard building block for the computations in the dynamic streams, is that our algorithm has to be deterministic here. Unfortunately, we have no idea how to obtain $N^-_t(v)$ for all v deterministically in the turnstile stream. To get around with this issue, we note that the nodes with $|N^-_t(v)| = t$ are not involved in the computation of the first t positions in σ_H, as shown in the proof of Lemma 15. We maintain the degree of node v using $O(1)$ space to decide whether v has $N^-_t(v) = t$ or not. If $N^-_t(v) = t$, the edges incident to v does not involve the computation of the first t positions in σ_H, so there is no need to compute $N^-_t(v)$; otherwise $N^-_t(v) < t$, we use Lemma 16 to recover $N^-_t(v)$ based on the characteristic polynomials used for the set reconciliation [19].

▶ **Lemma 16.** *For every integer $t \geq 1$, for every node $v \in V$ with $|N^-_t(v)| < t$, the set $N^-_t(v)$ can be obtained deterministically in the turnstile model using $O(t)$ space.*

Proof. Define the characteristic polynomial of v to be the univariate polynomial

$$P_v(Z) = \prod_{(u,v)\in N^-(v)} (Z-u) \pmod p \text{ for some prime } p > 2n,$$

noting that the roots of $P(Z)$ comprise the set $N^-(v)$ and the degree of $P_v(v)$ corresponds to $|N^-(v)|$. Though $|N_t^-(v)| < t$ at the end of the dynamic input stream, it can be much larger than t in the middle of processing the input. Hence, to recover $N_t^-(v)$ using $O(t)$ space we cannot directly maintain $P_v(Z)$. Instead, we maintain the values of $P_v(Z)$ at t distinct points $a_1, a_2, \dots, a_t$ outside the range $[1, n]$. That is, for each $i \in [1, t]$, while processing an edge insertion (u, v) update $P(a_i)$ with $P(a_i) \cdot (a_i - u)$ and while processing an edge deletion (u, v) update $P(a_i)$ with $P(a_i) \cdot (a_i - u)^{-1}$, i.e. the multiplicative inverse of $(a_i - u)$. Given the point-value pairs $\{(a_i, P(a_i)) : i \in [1, t]\}$, by the standard interpolation [19], if the degree of $P_v(Z)$ less than t at the end of the dynamic input stream (i.e. the given premise $|N_t^-(Z)| < t$) $P_v(Z)$ can be recovered. ◀

We are now ready to prove Theorem 3. We use $O(t)$-certificates n/t times, each requiring a single pass and using $O(tn)$ space. Set k as n/t, so our algorithm uses $O(n^2/k)$ space as claimed.

7 Applications

In this section, we present applications of our topological sorting algorithms. In Section 7.1, we develop an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm for SCC decomposition, based on the chain cover construction described in Section 3. We then present an $O(k)$-pass $O((n^2 \log n)/k)$-space streaming algorithm for SCC decomposition, using the notion of the t-certificate introduced in Section 6. Given these algorithms, the results for 2-SAT in Section 7.3 follow immediately.

7.1 An $O(1/\varepsilon)$-Pass $O(\alpha n^{1+\varepsilon})$-Space SCC decompisition Algorithm

In this subsection, we focus on the problem of SCC decomposition. Let $G = (V, E)$ be an n-node directed graph. A strongly connected component (SCC) $C \subseteq V$ is a maximal set of nodes such that, for all $u, v \in C$, there exists a directed path from u to v. The SCC decomposition of G is typically defined as a partition $T = \{C_1, C_2, \dots\}$ of V, where each C_i is an SCC of G.

However, in many applications, it is not only the identification of SCCs that matters, but also their topological ordering. For instance, the algorithm for finding a satisfying assignment of 2-SAT [2] relies on such an ordering.

Therefore, we define an SCC decomposition to be an ordered collection of sets $C_1, C_2, \dots, C_\ell$, where each C_i is an SCC, $\bigcup_{i\in[\ell]} C_i = V$, and there are no edges from any node in C_j to any node in C_i for all $i < j$.

We begin with showing that the transitive closure of an n-node directed graph G (possibly cyclic) with independence number α can be represented by the transitive closure of a subgraph $H \subseteq G$ containing $O(\alpha n)$ edges. We note that Lemma 17 extends Lemma 4, which applies only to DAGs.

▶ **Lemma 17.** *Every directed graph G (possibly cyclic) with independence number α contains an $O(\alpha n)$-edge subgraph H such that G and H have the same transitive closure. Moreover, if a graph G' has the same transitive closure as G, then G' also contains an $O(\alpha n)$-edge subgraph with the same transitive closure, even if the independence number of G' exceeds α.*

Proof. By the Gallai-Milgram Theorem [8], the node set of a directed graph with independence number α can be covered by at most α node-disjoint paths. The proof of Lemma 4 can be adapted to this setting by replacing the chain cover with the above α node-disjoint paths. ◀

▶ **Lemma 18.** *Let G be an n-node directed graph with independence number α. For every $\varepsilon > 0$, there exists an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm that outputs a collection of α node-disjoint directed paths whose node sets partition the node set of G.*

Proof. The algorithm is almost the same as that in Lemma 5 except that we replace the chain cover used in Lemma 5 with the path cover stated in Lemma 17. ◀

▶ **Corollary 19.** *Let G be an n-node directed graph with independence number α. For every $\varepsilon > 0$, there exists an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space streaming algorithm that outputs an $O(\alpha n)$-edge subgraph H such that G and H have the same transitive closure.*

Proof. By Lemma 18, we obtain a path cover of G consisting of α directed paths using the claimed number of passes and space. Then, in another pass over all edges of G, for each node x, we store all of its out-neighbors in a list. If the list exceeds α out-neighbors, at least one can be discarded, as established in the proof of Lemma 4. ◀

Finally, given the subgraph H, the SCC decomposition of G can be computed by computing the SCC decomposition of H. This gives an $O(1/\varepsilon)$-pass $O(\alpha n^{1+\varepsilon})$-space deterministic streaming algorithm to compute the SCC decomposition for any n-node directed graph G in the insertion-only model.

7.2 An $O(k)$-Pass $O((n^2 \log n)/k)$-Space SCC Decomposition Algorithm

In this subsection we devise an SCC decomposition algorithm based on our topological sort algorithm in Section 6. We first present an algorithm that w.h.p. identifies all SCCs that are large enough. We then demonstrate that a modified version of our topological sort algorithm can produce the SCC decomposition of G when all SCCs are small. Combining the two algorithms, we obtain an algorithm for SCC decomposition in directed graphs.

For every $v \in V$, let $R_{\text{out}}(v)$ be the set of nodes reachable from v, and $R_{\text{in}}(v)$ be the set of nodes that can reach v. The intersection $R_{\text{out}}(v) \cap R_{\text{in}}(v)$ is the SCC that includes v. We use this property to devise our algorithm. Given an integer $k \in [n]$, we sample a set S of $n \log n/k$ nodes from V uniformly at random independently. If we can compute $R_{\text{in}}(v) \cap R_{\text{out}}(v)$ for each $v \in S$, then all the SCCs containing at least k nodes are identified with high probability as shown in the following lemma.

▶ **Lemma 20.** *Let $G = (V, E)$ be an n-node directed graph. Given any $k \in [n]$, let S be a set of $\Theta(n \log n/k)$ nodes sampled uniformly at random from V. Every SCC of G that has at least k nodes contains some node in S with probability $1 - 1/n^{\Omega(1)}$.*

Proof. Let C be an SCC with at least k nodes. The probability that none of the nodes of C are sampled is at most $(1 - c \log n/k)^k \leq 1/n^c$ for some constant $c > 1$. Applying a union bound over all SCCs with at least k nodes, the probability that there exists such an SCC containing no sampled nodes is $1/n^{\Omega(1)}$. ◀

In what follows, we show how to compute $R_{\text{out}}(v) \cap R_{\text{in}}(v)$ for each $v \in S$. We perform BFS from each $v \in S$ in both forward and backward directions. Besides usual BFS steps, whenever a node $u \in S$ reaches another node $v \in S$ in the i-th forward (resp. backward)

BFS step, we merge all nodes that v can reach in $i-1$ forward (resp. backward) BFS steps to u. Formally, for all $v \in S$ and a given integer $i \in [n]$, let $R^i_{\text{out}}(v)$ be the set of nodes that v can reach in i BFS steps. $R^i_{\text{out}}(v)$ can be obtained from $R^{i-1}_{\text{out}}(v)$ within one pass. We initiate $R^i_{\text{out}}(v)$ to be $R^{i-1}_{\text{out}}(v)$. For each input edge (x, y), if $x \in R^{i-1}_{\text{out}}(v)$, then we add y to $R^i_{\text{out}}(v)$. If $y \in S$, then we also add $R^{i-1}_{\text{out}}(y)$ to $R^i_{\text{out}}(v)$. Before the algorithm starts, we initiate $R^0_{\text{out}}(v) = \{v\}$ for all $v \in S$. We define $R^i_{in}(v)$ similarly as above.

The following lemma shows that, with high probability, after k passes of forward and backward BFS from each sampled node v, the intersection $R_{\text{in}}(v) \cap R_{\text{out}}(v)$ is correctly identified for every sampled node v.

▶ **Lemma 21.** *Let $G = (V, E)$ be an n-node directed graph. Given any $k \in [n]$, let S be a set of $\Theta(n \log n/k)$ nodes sampled uniformly at random from V. Then, with probability $1 - 1/n^{\Omega(1)}$, $R_{out}(v) \cap R_{in}(v) = R^k_{out}(v) \cap R^k_{in}(v)$ for every $v \in S$.*

Proof. Consider a pair of nodes v, u where $v \in S$ and u is a node in the same SCC as v. By definition, there is a path from v to u and a path from u to v within the SCC. Let f be the length of a shortest path P from v to u. If $f \leq k$, apparently $u \in R^k_{\text{out}}(v)$. If $f \geq k$, we claim that in each subpath of length $k/2$ in P, there is a node $\in S$ with probaility at least $1 - 1/n^4$. Let $v, p_1, p_2, \cdots, p_w$ be the sampled nodes on the path arranged by their distance from v. By the claim, for all $i \in [w-1]$ the distance between p_i, p_{i+1} is at most k with probability at least $1 - 1/n^3$. Therefore $p_{i+1} \in R^k_{\text{out}}(p_i)$ for all $i \in [w-1]$. Also, by the claim, $u \in R^k_{\text{out}}(p_w)$ and $p_1 \in R^k_{\text{out}}(v)$. Observe that, by our construction, for every pair of $a, b \in S$, if $b \in R^k_{\text{out}}(a)$, then $R^k_{\text{out}}(b) \subseteq R^k_{\text{out}}(a)$. Therefore $u \in R^k_{\text{out}}(v)$. By a similar proof, $u \in R^k_{\text{in}}(v)$. Combining the two, we have $u \in R^k_{\text{out}}(v) \cap R^k_{\text{in}}(v)$ with probability at least $1 - 1/n^3$. Applying the union bound, we complete the proof. It remains to prove the claim. The probability that all $k/2$ nodes in a subpath are not sampled is at most $(1 - c \log n/k)^{k/2} = 1/n^4$ for a sufficiently large constant c, as desired. ◀

The above results lead to the following algorithm for finding all large enough SCCs.

▶ **Lemma 22.** *Let $G = (V, E)$ be an n-node directed graph. Given any $k \in [n]$, there is an $O(k)$ pass $O((n^2 \log n)/k)$ space streaming algorithm that determines all SCCs of G that contain at least k nodes with probability $1 - 1/n^{\Omega(1)}$.*

Proof. Given any k, the algorithm samples $\Theta(n \log n/k)$ nodes uniformly at random, and then performs BFS from each node for k steps in parallel. Lemma 21 shows that the reachability sets for all sampled nodes can be obtained with probability $1 - 1/n^{\Omega(1)}$, and Lemma 20 shows that all SCCs with at least k nodes will be found with probability $1 - 1/n^{\Omega(1)}$. The algorithm uses $O(k)$ passes to simulate the k-step BFS. Performing BFS from each of the $\Theta(n \log n/k)$ sampled nodes requires $O(n)$ space. Therefore, the space usage of the algorithm is $O((n^2 \log n)/k)$. ◀

It is unclear whether the algorithm described in Lemma 22 is optimal. Nevertheless, we present a one-pass lower bound by first proving a lower bound for an easier problem and then extend to it.

▶ **Theorem 23.** *Let $G = (V, E)$ be an n-node directed graph. Deciding whether G is strongly connected in one pass requires $\Omega(n^2)$ bits.*

Proof. A reduction from s, t-reachability to this problem is shown in [13] on a RAM. For completeness, we restate it here. Given an instance of s, t-reachability consisting of a directed graph $G = (V, E)$ and two designated nodes $s, t \in V$, we generate a new graph G' by adding edges (t, i) for all $i \in V \setminus \{t\}$ and (i, s) for all $i \in V \setminus \{s\}$ to G. G' is strongly connected if and only if G has a path from s to t. Adding the $2n$ edges is straightforward in the streaming model and hence the reduction can be done within $O(n)$ space and without using extra passes. Since s, t-reachability requires $\Omega(n^2)$ bits to solve in one pass [11], the same lower bound extends to this problem. ◀

Theorem 23 immediately implies the following:

▶ **Corollary 24.** *Given any integer $k \in [n]$, identifying all SCCs of G with at least k nodes in one pass requires $\Omega(n^2)$ bits.*

▶ **Corollary 25.** *Identifying all SCCs for an n-node directed graph G in one pass requires $\Omega(n^2)$ bits.*

In the rest of the section, we show that our topological sort algorithm from Section 6 can be modified to produce the SCC decomposition of G when all SCCs in G are small. We start with the following observation.

▶ **Lemma 26.** *Let G be an n-node directed graph, and let H be a source subgraph of G. For any SCC C of G, if C contains at least one node from H, then C is entirely contained within H.*

Proof. Suppose, for the sake of contradiction, that there exists an SCC C such that $C \cap H \neq \emptyset$ but $C \nsubseteq H$. That is, there exist nodes $u \in C$ where $u \notin H$. Take any $v \in C \cap H$, by the definition of an SCC, there must be a path from u to v. This implies the existence of an edge from $G \setminus H$ to H, which contradicts the assumption that H is a source subgraph. ◀

Using Lemma 26, we can design an algorithm for SCC decomposition. The process proceeds as follows. We first identify a source subgraph H of G and compute the SCC decomposition of H. We then remove H from G and repeat this procedure until we obtain the complete SCC decomposition of G.

Here we present an efficient way to find a source subgraph. We can assume that the input graph $G = (V, E)$ is an n-node directed graph with all SCCs having at most k nodes, as larger SCCs can be identified and contracted by the above algorithm. Given any $t \geq k$, we consider a subgraph

$$H_1 = (V, F) \text{ where } F := \bigcup_{v \in V} N^-_{2t}(v).$$

In the following lemmas, we will show that by leveraging the structure of H_1 along with the degree information of all nodes in G, we can identify a source subgraph containing at least t nodes.

▶ **Lemma 27.** *Let G be an n-node directed graph. Given $t \geq 1$, let Q be the set of nodes such that for all $v \in Q$, $|N^-(v)| \leq 2t$. Let*

$$H_1 = (V, F) \textit{ where } F := \bigcup_{v \in V} N^-_{2t}(v).$$

If H_2 is a source subgraph of H_1 and $V(H_2) \subseteq Q$, then H_2 is a source subgraph of G.

Proof. For all nodes $v \in Q$, since $|N^-(v)| \leq 2t$, we have $N^-(v) = N^-_{2t}(v)$. In other words, all nodes in Q have all their incoming edges recorded in H_1. Therefore, if $V(H_2) \subseteq Q$ has no incoming edges in H_1, it also has no incoming edges in G, which completes the proof. ◀

▶ **Lemma 28.** *Let G be an n-node directed graph with all SCCs having at most k nodes. Given any $t \in [k, n]$, let Q be the set of nodes such that for all $v \in Q$, $N^-(v) \leq 2t$. Let*

$$H_1 = (V, F) \text{ where } F := \bigcup_{v \in V} N^-_{2t}(v).$$

There exists a subgraph H_2 with at least t nodes s.t. H_2 is a source subgraph of H_1 and $V(H_2) \subseteq Q$.

Proof. Consider any topological ordering of the SCCs $C_1, C_2, \ldots, C_\ell$ of G. Let Δ be the largest integer in $[1, \ell]$ such that the number of nodes in the union of $C_1, C_2, \ldots, C_\Delta$ is at most $2t$. Define H_2 as the subgraph induced by the nodes in the union. We will show that this choice of H_2 satisfies all the required conditions.

Note that the SCCs $C_1, C_2, \ldots, C_\Delta$ have no incoming edges from $C_{\Delta+1}, C_{k+2}, \ldots, C_\ell$. Therefore, H_2 is a source subgraph of G and of any subgraph of G that contains it. Since every node contained in C_i for $i \leq \Delta$ has at most $2t$ incoming edges, each such node belongs to Q and the subgraph induced by the nodes in the union $C_1, C_2, \ldots, C_\Delta$ is contained in H_1. Thus, H_2 is a source subgraph of H_1. In addition, the union of $C_1, C_2, \ldots, C_{\Delta+1}$ must contain more than $2t$ nodes and $C_{\Delta+1}$ has at most k nodes, it follows that the union of $C_1, C_2, \ldots, C_\Delta$ contains at least $2t - k \geq t$ nodes, as required. ◀

By Lemmas 27 and 28, we get:

▶ **Theorem 29.** *Let G be an n-node directed graph such that all SCCs have at most k nodes. For any integer $t \geq k$, there is an $O(n/t)$-pass $O(tn)$-space streaming algorithm that determines the SCC decomposition of G.*

Proof. Combining Lemma 27 and Lemma 28, we can determine the SCC decomposition for at least t nodes by keeping $O(tn)$ edges in memory in one pass. We only need to repeat n/t times to obtain the SCC decomposition of G. The space requirement is $O(tn)$ since we keep at most $O(tn)$ edges in each pass, and maintaining the SCC decomposition uses $O(n)$ space. ◀

Combining Lemma 22 and Theorem 29, we obtain an SCC decomposition algorithm.

▶ **Theorem 30.** *Let G be an n-node directed graph. For any $k \leq \sqrt{n}$, there is an $O(k)$-pass $O((n^2 \log n)/k)$-space algorithm that determines the SCC decomposition of G with probability $1 - 1/n^{\Omega(1)}$ in the insertion-only model.*

Proof. We begin by applying the algorithm from Lemma 22 to G, which identifies all SCCs containing at least k nodes. This step runs in $O(k)$ passes and uses $O((n^2 \log n)/k)$ space, succeeding with probability at least $1 - 1/n^{\Omega(1)}$. After identifying these SCCs, we contract them, resulting in a new graph G', where all SCCs contain at most k nodes.

Next, we apply the algorithm from Theorem 29 to compute the SCC decomposition of G'. We set $t = n/k$, ensuring that $t \geq \sqrt{n} \geq k$, which satisfies the conditions of Theorem 29. Substituting these parameters, the total number of passes used is $O(k + n/t) = O(k)$, and the total space required is $O((n^2 \log n)/k + tn) = O((n^2 \log n)/k)$. ◀

7.3 2-SAT

In this subsection, we focus on the problem of 2-SAT. A 2-SAT instance F is a conjunction of m clauses, $F = \bigwedge_{i \in [m]} c_i$. Each clause $c_i = (\ell_i \vee r_i)$ is a disjunction of two literals. Every ℓ_i, r_i is one of the n Boolean variables $v_1, v_2, \ldots, v_n$ or their negations $\neg v_1, \neg v_2, \ldots, \neg v_n$. A satisfying assignment of F is an assignment of truth values for each variable v_i that makes F to be true, and we say F is satisfiable if there exists at least one satisfying assignment. The task is to determine if F is satisfiable, and if so, determine a satisfying assignment. Under streaming setting, each input of the input stream is a clause c_i, and the conjunction of all m inputs is the 2-SAT instance F under examination.

Before we show how to apply SCC decomposition to 2-SAT in streaming, we show a one pass lower bound for 2-SAT.

▶ **Theorem 31.** *Deciding if a 2SAT instance is satisfiable in one pass requires $\Omega(n^2)$ bits.*

Proof. We translate the reduction shown in [14] into streaming setting. Suppose to the contrary that we have a 2-SAT solver that could solve 2-SAT in one pass using $o(n^2)$ bits, then we have an algorithm that could solve s,t-reachability in one pass using $o(n^2)$ bits by the following reduction. Given an instance of s,t-reachability consisting of a directed graph $G = (V,E)$ and two designated nodes $s,t \in V$, for each input edge $(u,v) \in E$, generate input $(u \vee \neg v)$ for the 2-SAT solver, and for the two nodes s,t, generate input $(s \vee t), (\neg s \vee \neg t), (\neg s \vee t)$ for the 2-SAT solver. The 2SAT instance generated is unsatisfiable if and only if s can reach t in G. Since testing s,t-reachability in a directed graph in one pass requires $\Omega(n^2)$ bits [11], there cannot exist such a 2-SAT solver. ◀

Theorem 31 shows that, merely deciding if a 2-SAT instance is satisfiable in one pass requires space enough to keep the whole input. Henceforth we focus on multi-pass algorithms.

We follow the approach of [2] which reduces 2-SAT to SCC-decomposition. The reduction constructs an implication graph G from a 2-SAT instance F. The satisfiability and the truth assignment of F can be retrieved from the SCC decomposition of G. We show that, given an input stream of 2-SAT, the construction of a stream of its implication graph only double the input complexity. For each variable v and its negation $\neg v$, construct nodes $v, \neg v \in V$. For each input clause $c_i = (\ell_i \vee r_i)$, construct two edges $(\neg \ell_i, r_i)$, $(\neg r_i, \ell_i)$. The resulting implication graph G is a graph with $2n$ nodes and $2m$ edges. We can apply our SCC decomposition algorithms to G and obtain a solution to the 2-SAT instance.

▶ **Theorem 32.** *Given any $k \leq \sqrt{n}$, 2-SAT can be solved in $O(k)$-pass and $O((n^2 \log n)/k)$-space with probability $1 - 1/n^{\Omega(1)}$ in the insertion-only model.*

Let α_I denote the independence number of the implication graph of the input 2-SAT instance. We can apply our algorithm presented in Corollary 19.

▶ **Theorem 33.** *A 2-SAT instance F can be solved in $O(1/\varepsilon)$-pass $O(\alpha_I n^{1+\varepsilon})$-space in the insertion-only model.*

References

1 Akanksha Agrawal, Arindam Biswas, Édouard Bonnet, Nick Brettell, Radu Curticapean, Dániel Marx, Tillmann Miltzow, Venkatesh Raman, and Saket Saurabh. Parameterized streaming algorithms for Min-Ones d-SAT. In *39th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 150 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPICS.FSTTCS.2019.8`.

2 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. `doi:10.1016/0020-0190(79)90002-4`.

3 Béla Bollobás and Graham R. Brightwell. The dimension of random graph orders. In *The Mathematics of Paul Erdős II*, pages 47–68. Springer, 2013. doi:10.1007/978-1-4614-7254-4_5.

4 Manuel Cáceres. Minimum chain cover in almost linear time. In *50th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 31:1–31:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICALP.2023.31.

5 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1786–1802. SIAM, 2020. doi:10.1137/1.9781611975994.109.

6 Rajesh Chitnis and Graham Cormode. Towards a theory of parameterized streaming algorithms. In *14th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 148 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.IPEC.2019.7.

7 Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1234–1251. SIAM, 2015. doi:10.1137/1.9781611973730.82.

8 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

9 R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

10 Stefan Fafianie and Stefan Kratsch. Streaming kernelization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium (MFCS)*, volume 8635 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2014. doi:10.1007/978-3-662-44465-8_24.

11 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/J.TCS.2005.09.013.

12 Alan M. Frieze. On the independence number of random graphs. *Discret. Math.*, 81(2):171–175, 1990. doi:10.1016/0012-365X(90)90149-C.

13 Neil D. Jones. Space-bounded reducibility among combinatorial problems. *J. Comput. Syst. Sci.*, 11(1):68–85, 1975. doi:10.1016/S0022-0000(75)80050-X.

14 Neil D. Jones, Y. Edmund Lien, and William T. Laaser. New problems complete for nondeterministic log space. *Math. Syst. Theory*, 10:1–17, 1976. doi:10.1007/BF01683259.

15 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, (PODS)*, pages 49–58. ACM, 2011. doi:10.1145/1989284.1989289.

16 Shimon Kogan and Merav Parter. Faster and unified algorithms for diameter reducing shortcuts and minimum chain covers. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 212–239. SIAM, 2023. doi:10.1137/1.9781611977554.CH9.

17 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Meta-theorems for parameterized streaming algorithms. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 712–739. SIAM, 2024.

18 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

19 Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Trans. Inf. Theory*, 49(9):2213–2218, 2003. doi:10.1109/TIT.2003.815784.

20 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, August 2005. doi:10.1561/0400000002.

21 Jelle J. Oostveen and Erik Jan van Leeuwen. Parameterized complexity of streaming diameter and connectivity problems. *Algorithmica*, 86(9):2885–2928, 2024. `doi:10.1007/S00453-024-01246-Z`.

22 Warren Schudy. Finding strongly connected components in parallel using $o(\log^2 n)$ reachability queries. In *Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 146–151. ACM, 2008. `doi:10.1145/1378533.1378560`.