

# Average-Case Analysis of Online Topological Ordering

Deepak Ajwani and Tobias Friedrich

Max-Planck-Institut für Informatik, Saarbrücken, Germany

**Abstract.** Many applications like pointer analysis and incremental compilation require maintaining a topological ordering of the nodes of a directed acyclic graph (DAG) under dynamic updates. All known algorithms for this problem are either only analyzed for worst-case insertion sequences or only evaluated experimentally on random DAGs. We present the first average-case analysis of online topological ordering algorithms. We prove an expected runtime of  $\mathcal{O}(n^2 \text{polylog}(n))$  under insertion of the edges of a complete DAG in a random order for the algorithms of Alpern et al. (SODA, 1990), Katriel and Bodlaender (TALG, 2006), and Pearce and Kelly (JEA, 2006). This is much less than the best known worst-case bound  $\mathcal{O}(n^{2.75})$  for this problem.

## 1 Introduction

There has been a growing interest in dynamic graph algorithms over the last two decades due to their applications in a variety of contexts including operating systems, information systems, network management, assembly planning, VLSI design and graphical applications. Typical dynamic graph algorithms maintain a certain property (e. g., connectivity information) of a graph that changes (a new edge inserted or an existing edge deleted) dynamically over time. An algorithm or a problem is called *fully dynamic* if both edge insertions and deletions are allowed, and it is called *partially dynamic* if only one (either only insertion or only deletion) is allowed. If only insertions are allowed, the partially dynamic algorithm is called incremental; if only deletions are allowed, it is called decremental. While a number of fully dynamic algorithms have been obtained for various properties on undirected graphs (see [9] and references therein), the design and analysis of fully dynamic algorithms for directed graphs has turned out to be much harder (e. g., [12, 23–25]). Much of the research on directed graphs is therefore concentrated on the design of partially dynamic algorithms instead (e. g., [3, 6, 13]). In this paper, we focus on the analysis of algorithms for maintaining a topological ordering of directed graphs in an incremental setting.

A topological order  $T$  of a directed graph  $G = (V, E)$  (with  $n := |V|$  and  $m := |E|$ ) is a linear ordering of its nodes such that for all directed paths from  $x \in V$  to  $y \in V$  ( $x \neq y$ ), it holds that  $T(x) < T(y)$ . A directed graph has a topological ordering if and only if it is acyclic. There are well-known algorithms for computing the topological ordering of a directed acyclic graph (DAG) in

$\mathcal{O}(m + n)$  time in an offline setting (see e.g. [7]). In a fully dynamic setting, each time an edge is added or deleted from the DAG, we are required to update the bijective mapping  $T$ . In the online/incremental variant of this problem, the edges of the DAG are not known in advance but are inserted one at a time (no deletions allowed). As the topological order remains valid when removing edges, most algorithms for online topological ordering can also handle the fully dynamic setting. However, there are no good bounds known for the fully dynamic case. Most algorithms are only analyzed in the online setting.

Given an arbitrary sequence of edges, the online cycle detection problem is to discover the first edge which introduces a cycle. Till now, the best known algorithm for this problem involves maintaining an online topological order and returning the edge after which no valid topological order exists. Hence, results for online topological ordering also translate into results for the online cycle detection problem. Online topological ordering is required for incremental evaluation of computational circuits [2] and in incremental compilation [15, 17] where a dependency graph between modules is maintained to reduce the amount of re-compilation performed when an update occurs. An application for online cycle detection is pointer analysis [20].

The naïve way of computing an online topological order each time from scratch with the offline algorithm takes  $\mathcal{O}(m^2 + mn)$  time. Marchetti-Spaccamela, Nanni, and Rohnert [16] gave an algorithm that can insert  $m$  edges in  $\mathcal{O}(mn)$  time. Alpern, Hoover, Rosen, Sweeney, and Zadeck (AHR SZ) proposed an algorithm [2] which runs in  $\mathcal{O}(|\hat{K}| \log(|\hat{K}|))$  time per edge insertion with  $|\hat{K}|$  being a local measure of the insertion complexity. However, there is no analysis of AHR SZ for a sequence of edge insertions. Katriel and Bodlaender (KB) [13] analyzed a variant of the AHR SZ algorithm and obtained an upper bound of  $\mathcal{O}(\min\{m^{\frac{3}{2}} \log n, m^{\frac{3}{2}} + n^2 \log n\})$  for inserting an arbitrary sequence of  $m$  edges. The algorithm by Pearce and Kelly (PK) [18] empirically outperforms the other algorithms for random edge insertions leading to sparse random DAGs, although its worst-case runtime is inferior to KB. Recently, Ajwani, Friedrich, and Meyer (AFM) [1] proposed a new algorithm with runtime  $\mathcal{O}(n^{2.75})$ , which asymptotically outperforms KB on dense DAGs.

As noted above, the empirical performances on random edge insertion sequences (REIS) for the above algorithms are quite different from their worst-cases. While PK performs empirically better for REIS, KB and AFM are the best known algorithms for worst-case sequences. This leads us to the theoretical study of online topological ordering algorithms on REIS.

Our contributions are as follows:

- We show an expected runtime of  $\mathcal{O}(n^2 \log^2 n)$  for inserting all edges of a complete DAG in a random order with PK (cf. Section 4).
- For AHR SZ and KB, we show an expected runtime of  $\mathcal{O}(n^2 \log^3 n)$  for complete random edge insertion sequences (cf. Section 5). This is significantly better than the known worst-case bound of  $\mathcal{O}(n^3)$  for KB to insert  $\Omega(n^2)$  edges.

- Additionally, we show that for such edge insertions the expected number of edges which force any algorithm to change the topological order (“invalidating edges”) is  $\mathcal{O}(n^{\frac{3}{2}}\sqrt{\log n})$  (cf. Section 6), which is the first such result.

The remainder of this paper is organized as follows. The next section describes briefly the three algorithms AHRSZ, KB, and PK. In Section 3 we specify the random graph models used in our analysis. Sections 4-6 prove our upper bounds for the runtime of the three algorithms and the number of invalidating edges.

## 2 Algorithms

This section first introduces some notations and then describes the three algorithms AHRSZ, KB, and PK. We keep the current topological order as a bijective function  $T: V \rightarrow [1..n]$ . In this and the subsequent sections, we will use the following notations:  $d(u, v)$  denotes  $|T(u) - T(v)|$ ,  $u < v$  is a short form of  $T(u) < T(v)$ ,  $u \rightarrow v$  denotes an edge from  $u$  to  $v$ , and  $u \rightsquigarrow v$  expresses that  $v$  is reachable from  $u$ . Note that  $u \rightsquigarrow u$ , but *not*  $u \rightarrow u$ . The *degree* of a node is the sum of its in- and out-degree.

Consider the  $i$ -th edge insertion  $u \rightarrow v$ . We say that an edge insertion is *invalidating* if  $u > v$  before the insertion of this edge. We define  $R_B^{(i)} := \{x \in V \mid v \leq x \wedge x \rightsquigarrow u\}$ ,  $R_F^{(i)} := \{y \in V \mid y \leq u \wedge v \rightsquigarrow y\}$  and  $\delta^{(i)} = R_F^{(i)} \cup R_B^{(i)}$ . Let  $|\delta^{(i)}|$  denote the number of nodes in  $\delta^{(i)}$  and let  $\|\delta^{(i)}\|$  denote the number of edges incident to any node of  $\delta^{(i)}$ . Note that  $\delta^{(i)}$  as defined above is different from the adaptive parameter  $\delta$  of the bounded incremental computation model. If an edge is non-invalidating, then  $|R_B^{(i)}| = |R_F^{(i)}| = |\delta^{(i)}| = 0$ . Note that for an invalidating edge  $R_F^{(i)} \cap R_B^{(i)} = \emptyset$  as otherwise the algorithms will just report a cycle and terminate.

We now describe the insertion of the  $i$ -th edge  $u \rightarrow v$  for all the three algorithms. Assume for the remainder of this section that  $u \rightarrow v$  is an invalidating edge, as otherwise none of the algorithms do anything for that edge. We define an algorithm to be *local* if it only changes the ordering of nodes  $x$  with  $v \leq x \leq u$  to compute the new topological order  $T'$  of  $G \cup \{(u, v)\}$ . All three algorithms are local and they work in two phases - a “discovery phase” and a “relabelling phase”.

In the discovery phase of **PK**, the set  $\delta^{(i)}$  is identified using a forward depth-first search from  $v$  (giving a set  $R_F^{(i)}$ ) and a backward depth-first search from  $u$  (giving a set  $R_B^{(i)}$ ). The relabelling phase is also very simple. They sort both sets  $R_F^{(i)}$  and  $R_B^{(i)}$  separately in increasing topological order and then allocate new priorities according to the relative position in the sequence  $R_B^{(i)}$  followed by  $R_F^{(i)}$ . They do not alter the priority of any node not in  $\delta^{(i)}$ , thereby greatly simplifying the relabeling phase. The runtime of PK for a single edge insertion is  $\Theta(\|\delta^{(i)}\| + |\delta^{(i)}| \log |\delta^{(i)}|)$ .

Alpern et al. [2] used the bounded incremental computation model [23] and introduced the measure  $|\hat{K}|$ . For an invalidated topological order  $T$ , the set

$K \subseteq V$  is a *cover* if for all  $x, y \in V$ :  $(x \rightsquigarrow y \wedge y < x \Rightarrow x \in K \vee y \in K)$ . This states that for any connected  $x$  and  $y$  which are incorrectly ordered, a cover  $K$  must include  $x$  or  $y$  or both.  $|K|$  and  $\|K\|$  denote the number of nodes and edges touching nodes in  $K$ , respectively. We define  $\|K\| := |K| + \|K\|$  and a cover  $\hat{K}$  to be *minimal* if  $\|\hat{K}\| \leq \|K\|$  for any other cover  $K$ . Thus,  $\|\hat{K}\|$  captures the minimal amount of work required to calculate the new topological order  $T'$  of  $G \cup \{(u, v)\}$  assuming that the algorithm is local and that the adjacent edges must be traversed.

**AHRSZs discovery phase** marks the nodes of a cover  $K$  by marking some of the unmarked nodes  $x, y \in \delta^{(i)}$  with  $x \rightsquigarrow y$  and  $y < x$ . This is done recursively by moving two frontiers starting from  $v$  and  $u$  towards each other. Here, the crucial decision is which frontier to move next. AHRSZ tries to minimize  $\|K\|$  by balancing the number of edges seen on both sides of the frontier. The recursion stops when forward and backward frontier meet. Note that we do not necessarily visit all nodes in  $R_F^{(i)}$  ( $R_B^{(i)}$ ) while extending the forward frontier (backward frontier). It can be proven that the marked nodes indeed form a cover  $K$  and that  $\|K\| \leq 3\|\hat{K}\|$ .

The *relabeling phase* employs the dynamic priority space data structure due to Dietz and Sleator [8]. This permits new priorities to be created between existing ones in  $\mathcal{O}(1)$  amortized time. This is done in two passes over the nodes in  $K$ . During the first pass, it visits the nodes of  $K$  in reverse topological order and computes a strict upper bound on the new priorities to be assigned to each node. In the second phase, it visits the nodes in  $K$  in topological order and computes a strict lower bound on the new priorities. Both together allow to assign new priorities to each node in  $K$ . Thereafter they minimize the number of different labels used to speed up the operations on the priority space data structure in practice. It can be proven that the discovery phase with  $\|\hat{K}\|$  priority queue operations dominates the time complexity, giving an overall bound of  $\mathcal{O}(\|\hat{K}\| \log \|K\|)$ .

**KB** is a slight modification of AHRSZ. In the discovery phase AHRSZ counts the total number of edges incident on a node. KB counts instead only the in-degree of the backward frontier nodes and only the out-degree of the forward frontier nodes. In addition, KB also simplified the relabeling phase. The nodes visited during the extension of the forward (backward) frontier are deleted from the dynamic priority space data-structure and are reinserted, in the same relative order among themselves, after (before) all nodes in  $R_B^{(i)}$  ( $R_F^{(i)}$ ) not visited during the backward (forward) frontier extension. The algorithm thus computes a cover  $K \subseteq \delta^{(i)}$  and its complexity per edge insertion is  $\mathcal{O}(\|K\| \log \|K\|)$ . The worst case running time of KB for a sequence of  $m$  edge insertions is  $\mathcal{O}(\min\{m^{\frac{3}{2}} \log n, m^{\frac{3}{2}} + n^2 \log n\})$ .

### 3 Random Graph Model

Erdős and Rényi [10, 11] introduced and popularized random graphs. They defined two closely related models:  $G(n, p)$  and  $G(n, M)$ . The  $G(n, p)$  model

( $0 < p < 1$ ) consists of a graph with  $n$  nodes in which each edge is chosen independently with probability  $p$ . On the other hand, the  $G(n, M)$  model assigns equal probability to all graphs with  $n$  nodes and exactly  $M$  edges. Each such graph occurs with a probability of  $1/\binom{N}{M}$ , where  $N := \binom{n}{2}$ .

For our study of online topological ordering algorithms, we use the random DAG model of Barak and Erdős [4]. They obtain a random DAG by directing the edges of an undirected random graph from lower to higher indexed vertices. Depending on the underlying random graph model, this defines the  $DAG(n, p)$  and  $DAG(n, M)$  model. We will mainly work on the  $DAG(n, M)$  model since it is better suited to describe incremental addition of edges.

The set of all DAGs with  $n$  nodes is denoted by  $DAG^n$ . For a random variable  $f$  with probability space  $DAG^n$ ,  $\mathbf{E}_M(f)$  and  $\mathbf{E}_p(f)$  denotes the expected value in the  $DAG(n, M)$  and  $DAG(n, p)$  model, respectively. For the remainder of this paper, we set  $\mathbf{E}(f) := \mathbf{E}_M(f)$  and  $q := 1 - p$ .

The following theorem shows that in most investigations the models  $DAG(n, p)$  and  $DAG(n, M)$  are practically interchangeable, provided  $M$  is close to  $pN$ .

**Theorem 1** *Given a function  $f: DAG^n \rightarrow [0, a]$  with  $a > 0$  and  $f(G) \leq f(H)$  for all  $G \subseteq H$  and functions  $p$  and  $M$  of  $n$  with  $0 < p < 1$  and  $M \in \mathbb{N}$ .*

1. *If  $\lim_{n \rightarrow \infty} pqN = \lim_{n \rightarrow \infty} \frac{pN - M}{\sqrt{pqN}} = \infty$ , then  $\mathbf{E}_M(f) \leq \mathbf{E}_p(f) + o(1)$ .*
2. *If  $\lim_{n \rightarrow \infty} pqN = \lim_{n \rightarrow \infty} \frac{M - pN}{\sqrt{pqN}} = \infty$ , then  $\mathbf{E}_p(f) \leq \mathbf{E}_M(f) + o(1)$ .*

The analogous theorem for the undirected graph models  $G(n, p)$  and  $G(n, M)$  is well known. A closer look at the proof for it given by Bollobás [5] reveals that the probabilistic argument used to show the close connection between  $G(n, p)$  and  $G(n, M)$  can be applied in the same manner for the two random DAG models  $DAG(n, p)$  and  $DAG(n, M)$ .

We define a random edge sequence to be a uniform random permutation of the edges of a complete DAG, i. e., all permutations of  $\binom{n}{2}$  edges are equally likely. If the edges appear to the online algorithm in the order in which they appear in the random edge sequence, we call it a random edge insertion sequence (REIS). Note that a DAG obtained after inserting  $M$  edges of a REIS will have the same probability distribution as  $DAG(n, M)$ . To simplify the proofs, we first show our results in  $DAG(n, p)$  model and then transfer them in the  $DAG(n, M)$  model by Theorem 1.

## 4 Analysis of PK

When inserting the  $i$ -th edge  $u \rightarrow v$ , PK only regards nodes in  $\delta^{(i)} := \{x \in V \mid v \leq x \leq u \wedge (v \rightsquigarrow x \vee x \rightsquigarrow u)\}$  with “ $\leq$ ” defined according to the current topological order. As discussed in Section 2, PK performs  $\mathcal{O}(\|\delta^{(i)}\| + |\delta^{(i)}| \log |\delta^{(i)}|)$  operations for inserting the  $i$ -th edge. Theorems 4 and 10 of this section show for a random edge insertion sequence (REIS) of  $N$  edges that  $\sum_{i=1}^N |\delta^{(i)}| = \mathcal{O}(n^2)$  and  $\mathbf{E}(\sum_{i=1}^N \|\delta^{(i)}\|) = \mathcal{O}(n^2 \log^2 n)$ . This proves the following theorem.

**Theorem 2** For a random edge insertion sequence (REIS) leading to a complete DAG, the expected runtime of PK is  $\mathcal{O}(n^2 \log^2 n)$ .

A comparable pair (of nodes) are two distinct nodes  $x$  and  $y$  such that either  $x \rightsquigarrow y$  or  $y \rightsquigarrow x$ . We define a potential function  $\Phi_i$  similar to Katriel and Bodlaender [13]. Let  $\Phi_i$  be the number of comparable pairs after the insertion of  $i$  edges. Clearly,

$$\begin{aligned} \Delta\Phi_i &:= \Phi_i - \Phi_{i-1} \geq 0 \quad \text{for all } 1 \leq i \leq M, \\ \Phi_0 &= 0, \quad \text{and} \quad \Phi_M \leq n(n-1)/2. \end{aligned} \tag{1}$$

**Theorem 3** For all edge sequences, (i)  $|\delta^{(i)}| \leq \Delta\Phi_i + 1$  and (ii)  $|\delta^{(i)}| \leq 2\Delta\Phi_i$ .

*Proof.* Consider the  $i$ -th edge  $(u, v)$ . If  $u < v$ , the theorem is trivial since  $|\delta^{(i)}| = 0$ . Otherwise, each vertex of  $R_F^{(i)}$  and  $R_B^{(i)}$  (as defined in Section 2) gets newly ordered with respect to  $u$  and  $v$ , respectively. The set  $\bigcup_{x \in R_B^{(i)}} (x, v) \cap \bigcup_{x \in R_F^{(i)}} (u, x) = \{(u, v)\}$  as otherwise it will imply a discovered cycle and the algorithm will report the cycle and terminate. This means that overall at least  $|R_F^{(i)}| + |R_B^{(i)}| - 1$  node pairs get newly ordered:

$$\Delta\Phi_i \geq |R_F^{(i)}| + |R_B^{(i)}| - 1 = |\delta^{(i)}| - 1.$$

Also, since in this case  $\Delta\Phi_i \geq 1$ ,  $|\delta^{(i)}| \leq 2\Delta\Phi_i$ .  $\square$

**Theorem 4** For all edge sequences,  $\sum_{i=1}^N |\delta^{(i)}| \leq n(n-1) = \mathcal{O}(n^2)$ .

*Proof.* By Theorem 3 (i), we get  $\sum_{i=1}^N |\delta^{(i)}| \leq \sum_{i=1}^N (\Delta\Phi_i + 1) = \Phi_N + N \leq n(n-1)/2 + n(n-1)/2 = n(n-1)$ .  $\square$

The remainder of this section provides the necessary tools step by step to finally prove the desired bound on  $\sum_{i=1}^N \|\delta^{(i)}\|$  in Theorem 10. One can also interpret  $\Phi_i$  as a random variable in  $DAG(n, M)$  with  $M = i$ . The corresponding function  $\Psi$  for  $DAG(n, p)$  is defined as the total number of comparable node pairs in  $DAG(n, p)$ . Pittel and Tungol [21] showed the following theorem.

**Theorem 5** For  $p := c \log(n)/n$  and  $c > 1$ ,  $\mathbf{E}_p(\Psi) = (1 + o(1)) \frac{n^2}{2} \left(1 - \frac{1}{c}\right)^2$ .

Using Theorem 1, this result can be transformed to  $\Phi$  as defined above for  $DAG(n, M)$  and gives the following bounds for  $\mathbf{E}_M(\Phi_k)$ .

**Theorem 6** For  $n \log n < k \leq N - 2n \log n$ ,

$$\mathbf{E}_M(\Phi_k) = (1 + o(1)) \frac{n^2}{2} \left(1 - \frac{(n-1) \log n}{2(k + n \log n)}\right)^2.$$

For  $N - 2n \log n < k \leq N - 2 \log n$ ,

$$\mathbf{E}_M(\Phi_k) = (1 + o(1)) \frac{n^2}{2} \left(1 - \frac{(n-1) \log n}{2(k + \sqrt{n(N-k)})}\right)^2.$$

The formal proof of the above theorem will be given in the full version of the paper.

The degree sequence of a random graph is a well-studied problem. The following theorem is shown in [5].

**Theorem 7** *If  $pn/\log n \rightarrow \infty$ , then almost every graph  $G$  in the  $G(n, p)$  model satisfies  $\Delta(G) = (1 + o(1))pn$ , where  $\Delta(G)$  is the maximum degree of a node in  $G$ .*

As noted in Section 3, the undirected graph obtained by ignoring the directions of  $DAG(n, p)$  is a  $G(n, p)$  graph. Therefore, the above result is also true for the maximum degree (in-degree + out-degree) of a node in  $DAG(n, p)$ . Using Theorem 1, the above result can be transformed to  $DAG(n, M)$ , as well.

**Theorem 8** *With probability  $1 - \mathcal{O}(1/n)$ , there is no node with degree higher than  $c\frac{M}{n}$  for  $n \geq n_0$  and  $M > n \log n$  in  $DAG(n, M)$ , where  $c$  and  $n_0$  are fixed constants.*

The formal proof for  $c = 9$  will be given in the full version of the paper.

As the maximum degree of a node in  $DAG(n, i)$  is  $\mathcal{O}(i/n)$ , we finally just need to show a bound on  $\sum_i (i \cdot |\delta^{(i)}|)$  to prove Theorem 10. This is done in the following theorem.

**Theorem 9** *For  $DAG(n, M)$  and  $r := N - 2 \log n$ ,*

$$\mathbf{E}\left(\sum_{i=1}^r (i \cdot |\delta^{(i)}|)\right) = \mathcal{O}(n^3 \log^2 n).$$

*Proof.* Let us decompose the analysis in three steps. First, we show a bound on the first  $n \log n$  edges. By definition of  $\delta^{(i)}$ ,  $|\delta^{(i)}| \leq n$ . Therefore,

$$\sum_{i=1}^{n \log n} i \cdot \mathbf{E}(|\delta^{(i)}|) \leq \sum_{i=1}^{n \log n} i \cdot n = \mathcal{O}(n^3 \log^2 n). \quad (2)$$

The second step is to bound  $\sum_{i=n \log n}^t i \cdot |\delta^{(i)}|$  with  $t := N - 2n \log n$ . For this, Theorem 3 (ii) shows for all  $k$  such that  $n \log n < k < t$  that

$$\mathbf{E}\left(\sum_{i=k}^t |\delta^{(i)}|\right) \leq 2 \mathbf{E}\left(\sum_{i=k}^t \Delta \Phi_i\right) = 2 \mathbf{E}(\Phi_t - \Phi_{k-1}) = 2 \mathbf{E}(\Phi_t) - 2 \mathbf{E}(\Phi_{k-1}).$$

Using Theorem 6 and the fact that the hidden functions of the  $o(1)$  are decreasing in  $p$  [21], this yields (with  $s := n \log n$ )

$$\begin{aligned}
\mathbf{E}\left(\sum_{i=k}^t |\delta^{(i)}|\right) &\leq (1+o(1))n^2\left(\left(1-\frac{(n-1)\log n}{2(t+s)}\right)^2 - \left(1-\frac{(n-1)\log n}{2(k-1+s)}\right)^2\right) \\
&= (1+o(1))n^2(n-1)\log n\left(\frac{2}{2(k-1+s)} - \frac{2}{2(t+s)} + \right. \\
&\quad \left.\frac{(n-1)\log n}{4}\left(\frac{1}{(t+s)^2} - \frac{1}{(k-1+s)^2}\right)\right) \\
&\leq (1+o(1))n^2(n-1)\log n\left(\frac{1}{k-1+s} - \frac{1}{t+s}\right) \\
&\leq (1+o(1))n^2(n-1)\log n\frac{1}{k-1}. \tag{3}
\end{aligned}$$

By linearity of expectation and Equation (3),

$$\begin{aligned}
\mathbf{E}\left(\sum_{i=s+1}^t i|\delta^{(i)}|\right) &= \sum_{i=s+1}^t \left(i\mathbf{E}(|\delta^{(i)}|)\right) \leq \sum_{j=1}^{\log(\lceil \frac{t}{s} \rceil)} \left(2^j s \sum_{i=2^{(j-1)}s+1}^{2^j s} \mathbf{E}(|\delta^{(i)}|)\right) \\
&\leq \sum_{j=1}^{\log(\lceil \frac{t}{s} \rceil)} \left(2^j s \sum_{i=2^{(j-1)}s+1}^t \mathbf{E}(|\delta^{(i)}|)\right) \\
&\leq \sum_{j=1}^{\log(\lceil \frac{t}{s} \rceil)} \left(2^j s(1+o(1))n^2(n-1)\log n\frac{1}{2^{(j-1)}s}\right) \\
&= \sum_{j=1}^{\log(\lceil \frac{t}{s} \rceil)} \left(2(1+o(1))n^2(n-1)\log n\right) \\
&= 2(1+o(1))n^2(n-1)\log^2 n = \mathcal{O}(n^3 \log^2 n).
\end{aligned}$$

For the last step consider a  $k$  such that  $t < k < r$ . Theorem 3 (ii) gives

$$\mathbf{E}\left(\sum_{i=k}^r |\delta^{(i)}|\right) \leq 2\mathbf{E}\left(\sum_{i=k}^r \Delta\Phi_i\right) = 2\mathbf{E}(\Phi_r - \Phi_{k-1}) = 2\mathbf{E}(\Phi_r) - 2\mathbf{E}(\Phi_{k-1}).$$

Using Theorem 6 and similar arguments as before, this yields (with  $s(k) := \sqrt{\log n (N-k)}$ )

$$\begin{aligned}
\mathbf{E}\left(\sum_{i=k}^r |\delta^{(i)}|\right) &\leq (1+o(1))n^2\left(\left(1-\frac{(n-1)\log n}{2(r+s(r))}\right)^2 - \left(1-\frac{(n-1)\log n}{2(k-1+s(k-1))}\right)^2\right) \\
&= (1+o(1))n^2(n-1)\log n\left(\frac{2}{2(k-1+s(k-1))} - \frac{2}{2(r+s(r))} + \right. \\
&\quad \left.\frac{(n-1)\log n}{4}\left(\frac{1}{(r+s(r))^2} - \frac{1}{(k-1+s(k-1))^2}\right)\right).
\end{aligned}$$

Since  $k + s(k)$  is monotonically increasing for  $t < k < r$ ,  $\frac{1}{(k+s(k))^2}$  is a monotonically decreasing function in this interval. Therefore,  $\frac{1}{(r+s(r))^2} - \frac{1}{(k-1+s(k-1))^2} < 0$ , which proves the following equation.

$$\begin{aligned} \mathbf{E}\left(\sum_{i=k}^r |\delta^{(i)}|\right) &\leq (1 + o(1)) n^2 (n-1) \log n \left( \frac{1}{k-1+s(k-1)} - \frac{1}{r+s(r)} \right) \\ &\leq (1 + o(1)) n^2 (n-1) \log n \frac{1}{k-1}. \end{aligned} \quad (4)$$

By linearity of expectation and Equation (4),

$$\begin{aligned} \mathbf{E}\left(\sum_{i=N-2n \log n+1}^r i |\delta^{(i)}|\right) &= \sum_{i=N-2n \log n+1}^r \left( i \mathbf{E}(|\delta^{(i)}|) \right) \\ &\leq (N - 2 \log n) \sum_{i=N-2n \log n+1}^r \mathbf{E}(|\delta^{(i)}|) \\ &\leq (N - 2 \log n) (1 + o(1)) n^2 (n-1) \log n \frac{1}{N - 2n \log n - 1} \\ &= \mathcal{O}(n^3 \log n). \quad \square \end{aligned}$$

**Theorem 10** For  $\text{DAG}(n, M)$ ,  $\mathbf{E}\left(\sum_{i=1}^N \|\delta^{(i)}\|\right) = \mathcal{O}(n^2 \log^2 n)$ .

*Proof.* By definition of  $\|\delta^{(i)}\|$ , we know  $\|\delta^{(i)}\| \leq i$  and hence

$$\sum_{i=1}^{n \log n} \|\delta^{(i)}\| = \mathcal{O}(n^2 \log^2 n).$$

Again, let  $r := N - 2 \log n$ . Theorem 8 tells us that with probability greater than  $(1 - \frac{c'}{n})$  for some constant  $c'$ , there is no node with degree  $\geq \frac{c'}{n}$ . Since the degree of an arbitrary node in a DAG is bounded by  $n$ , we get with Theorems 4 and 9,

$$\begin{aligned} \mathbf{E}\left(\sum_{i=n \log n+1}^r \|\delta^{(i)}\|\right) &= \mathcal{O}\left(\mathbf{E}\left(\sum_{i=n \log n+1}^r \frac{c' i |\delta^{(i)}|}{n}\right) + \mathbf{E}\left(\sum_{i=n \log n+1}^r \frac{n c' |\delta^{(i)}|}{n}\right)\right) \\ &= \mathcal{O}\left(\frac{1}{n} \mathbf{E}\left(\sum_{i=1}^r (i |\delta^{(i)}|)\right) + n^2\right) \\ &= \mathcal{O}\left(\frac{1}{n} (n^3 \log^2 n) + n^2\right) = \mathcal{O}(n^2 \log^2 n). \end{aligned}$$

By again using the fact that the degree of an arbitrary node in a DAG is at most  $n$ , we obtain

$$\mathbf{E}\left(\sum_{i=r+1}^N \|\delta^{(i)}\|\right) = \mathcal{O}\left(n \cdot \mathbf{E}\left(\sum_{i=r+1}^N |\delta^{(i)}|\right)\right) = \mathcal{O}\left(n \cdot \sum_{i=r+1}^N n\right) = \mathcal{O}(n^2 \log n).$$

Thus,

$$\begin{aligned} \mathbf{E}\left(\sum_{i=1}^N \|\delta^{(i)}\|\right) &= \mathbf{E}\left(\sum_{i=1}^{n \log n} \|\delta^{(i)}\|\right) + \mathbf{E}\left(\sum_{i=n \log n+1}^r \|\delta^{(i)}\|\right) + \mathbf{E}\left(\sum_{i=r+1}^N \|\delta^{(i)}\|\right) \\ &= \mathcal{O}(n^2 \log^2 n) + \mathcal{O}(n^2 \log^2 n) + \mathcal{O}(n^2 \log n) = \mathcal{O}(n^2 \log^2 n). \quad \square \end{aligned}$$

## 5 Analysis of AHRSZ and KB

Katriel and Bodlaender [13] introduced KB as a variant of AHRSZ for which a worst-case runtime of  $\mathcal{O}(\min\{m^{\frac{3}{2}} \log n, m^{\frac{3}{2}} + n^2 \log n\})$  can be shown. In this section, we prove an expected runtime of  $\mathcal{O}(n^2 \log^3 n)$  under random edge insertion sequences, both for AHRSZ and KB.

Recall from Section 2 that for every edge insertion there is a minimal cover  $\hat{K}^{(i)}$ . In appendix C, we show that  $\delta^{(i)}$  is also a valid cover in this situation. Therefore, by definition of  $|\hat{K}^{(i)}|$ ,  $|\hat{K}^{(i)}| \leq |\delta^{(i)}| = |\delta^{(i)}| + \|\delta^{(i)}\|$ .

$$\mathbf{E}\left(\sum_{i=1}^m |\hat{K}^{(i)}|\right) \leq \sum_{i=1}^m |\delta^{(i)}| + \mathbf{E}\left(\sum_{i=1}^m \|\delta^{(i)}\|\right) = \mathcal{O}(n^2 \log^2 n)$$

The latter equality follows from Theorems 4 and 10. The expected complexity of AHRSZ on REIS is thus  $\mathcal{O}(\mathbf{E}(\sum_{i=1}^m |\hat{K}^{(i)}| \log n)) = \mathcal{O}(n^2 \log^3 n)$ .

KB also computes a cover  $K \subseteq \delta^{(i)}$  and its complexity per edge insertion is  $\mathcal{O}(|K| \log |K|)$ . Therefore,  $|K| \leq |\delta^{(i)}| + \|\delta^{(i)}\|$  and with a similar argument as above, the expected complexity of KB on REIS is  $\mathcal{O}(n^2 \log^3 n)$ .

## 6 Bounding the number of invalidating edges

An interesting question in all this analysis is how many edges will actually invalidate the topological ordering and force any algorithm to do something about them. Here, we show a non-trivial upper bound on the expected value of the number of invalidating edges on REIS. Consider the following random variable:  $\text{INVAL}(i) = 1$  if the  $i$ -th edge inserted is an invalidating edge;  $\text{INVAL}(i) = 0$  otherwise.

**Theorem 11**  $\mathbf{E}\left(\sum_{i=1}^m \text{INVAL}(i)\right) = \mathcal{O}(\min\{m, n^{\frac{3}{2}} \log^{\frac{1}{2}} n\})$ .

*Proof.* If the  $i$ -th edge is invalidating,  $|\delta^{(i)}| \geq 2$ ; otherwise  $\text{INVAL}(i) = |\delta^{(i)}| = 0$ . In either case,  $\text{INVAL}(i) \leq |\delta^{(i)}|/2$ . Thus, for  $s := n^{\frac{3}{2}} \log^{\frac{1}{2}} n$  and  $t := \min\{m, N - 2n \log n\}$ ,

$$\mathbf{E}\left(\sum_{i=s+1}^t \text{INVAL}(i)\right) \leq \mathbf{E}\left(\sum_{i=s+1}^t \frac{|\delta^{(i)}|}{2}\right) \leq \frac{(1 + o(1))}{2} n^{\frac{3}{2}} \log^{\frac{1}{2}} n.$$

The second inequality follows by substituting  $k := s + 1$  in Equation (3). Also, since the number of invalidating edges can be at most equal to the total number of edges,  $\sum_{i=1}^s \text{INVAL}(i) \leq s$ .

$$\begin{aligned} \mathbf{E}\left(\sum_{i=1}^m \text{INVAL}(i)\right) &= \mathbf{E}\left(\sum_{i=1}^s \text{INVAL}(i)\right) + \mathbf{E}\left(\sum_{i=s+1}^t \text{INVAL}(i)\right) + \mathbf{E}\left(\sum_{i=t}^m \text{INVAL}(i)\right) \\ &\leq \mathcal{O}(s) + \mathcal{O}(n^{\frac{3}{2}} \log^{\frac{1}{2}} n) + \mathcal{O}(n \log n) = \mathcal{O}(n^{\frac{3}{2}} \log^{\frac{1}{2}} n). \end{aligned}$$

The second bound  $\mathbf{E}(\sum_{i=1}^m \text{INVAL}(i)) \leq m$  is obvious by definition of  $\text{INVAL}(i)$ .  $\square$

## 7 Discussion

On random edge insertion sequences (REIS) leading to a complete DAG, we have shown an expected runtime of  $\mathcal{O}(n^2 \log^2 n)$  for PK and  $\mathcal{O}(n^2 \log^3 n)$  for AHSZ and KB while the trivial lower bound is  $\Omega(n^2)$ . Extending the average case analysis for the case where we only insert  $m$  edges with  $m \ll n^2$  still remains open. On the other hand, the only non-trivial lower bound for this problem is by Ramalingam and Reps [22], who have shown that an adversary can force any algorithm which maintains explicit labels to require  $\Omega(n \log n)$  time complexity for inserting  $n - 1$  edges. There is still a large gap between the lower bound of  $\Omega(\max\{n \log n, m\})$ , the best average-case bound of  $\mathcal{O}(n^2 \log^2 n)$  and the worst-case bound of  $\mathcal{O}(\min\{m^{1.5} + n^2 \log n, m^{1.5} \log n, n^{2.75}\})$ . Bridging this gap remains an open problem.

## Acknowledgements

The authors are grateful to Telikepalli Kavitha, Irit Katriel, and Ulrich Meyer for various helpful discussions.

## References

- [1] D. Ajwani, T. Friedrich, and U. Meyer. An  $\mathcal{O}(n^{2.75})$  algorithm for online topological ordering. In *Proc. of SWAT '06*, Vol. 4059 of *LNCS*, pp. 53–64, 2006.
- [2] B. Alpern, R. Hoover, B. K. Rosen, P. F. Sweeney, and F. K. Zadeck. Incremental evaluation of computational circuits. In *Proc. of SODA '90*, pp. 32–42, 1990.
- [3] G. Ausiello, G. F. Italiano, A. Marchetti-Spaccamela, and U. Nanni. Incremental algorithms for minimal length paths. *J. Algorithms*, 12(4):615–638, 1991.
- [4] A. B. Barak and P. Erdős. On the maximal number of strongly independent vertices in a random acyclic directed graph. *SIAM Journal on Algebraic and Discrete Methods*, 5(4):508–514, 1984.
- [5] B. Bollobás. *Random Graphs*. Cambridge Univ. Press, 2001.
- [6] S. Cicerone, D. Frigioni, U. Nanni, and F. Pugliese. A uniform approach to semi-dynamic problems on digraphs. *Theor. Comput. Sci.*, 203(1):69–90, 1998.

- [7] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1989.
- [8] P. F. Dietz and D. D. Sleator. Two algorithms for maintaining order in a list. In *Proc. of STOC '87*, pp. 365–372, 1987.
- [9] D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. In M. J. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 8. CRC Press, 1999.
- [10] P. Erdős and A. Rényi. On random graphs. *Publ Math Debrecen*, 6:290–297, 1959.
- [11] P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutato Int. Kozl.*, 5:17–61, 1960.
- [12] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic shortest paths and negative cycles detection on digraphs with arbitrary arc weights. In *Proc. of ESA '98*, Vol. 1461 of *LNCS*, pp. 320–331, 1998.
- [13] I. Katriel and H. L. Bodlaender. Online topological ordering. *ACM Trans. Algorithms*, 2(3):364–379, 2006. Preliminary version appeared as [14].
- [14] I. Katriel and H. L. Bodlaender. Online topological ordering. In *Proc. of SODA '05*, pp. 443–450, 2005.
- [15] A. Marchetti-Spaccamela, U. Nanni, and H. Rohnert. On-line graph algorithms for incremental compilation. In *Proc. of WG '93*, Vol. 790 of *LNCS*, pp. 70–86, 1993.
- [16] A. Marchetti-Spaccamela, U. Nanni, and H. Rohnert. Maintaining a topological order under edge insertions. *Information Processing Letters*, 59(1):53–58, 1996.
- [17] S. M. Omohundro, C.-C. Lim, and J. Bilmes. The sather language compiler/debugger implementation. Technical Report 92-017, International Computer Science Institute, Berkeley, 1992.
- [18] D. J. Pearce and P. H. J. Kelly. A dynamic topological sort algorithm for directed acyclic graphs. *J. Exp. Algorithmics*, 11:1.7, 2006. Preliminary version appeared as [19].
- [19] D. J. Pearce and P. H. J. Kelly. A dynamic algorithm for topologically sorting directed acyclic graphs. In *Proc. of WEA '04*, Vol. 3059 of *LNCS*, pp. 383–398, 2004.
- [20] D. J. Pearce, P. H. J. Kelly, and C. Hankin. Online cycle detection and difference propagation: Applications to pointer analysis. *Software Quality Journal*, 12(4):311–337, 2004.
- [21] B. Pittel and R. Tugol. A phase transition phenomenon in a random directed acyclic graph. *Random Struct. Algorithms*, 18(2):164–184, 2001.
- [22] G. Ramalingam and T. W. Reps. On competitive on-line algorithms for the dynamic priority-ordering problem. *Information Processing Letters*, 51:155–161, 1994.
- [23] G. Ramalingam and T. W. Reps. On the computational complexity of dynamic graph problems. *Theor. Comput. Sci.*, 158(1–2):233–277, 1996.
- [24] L. Roditty and U. Zwick. A fully dynamic reachability algorithm for directed graphs with an almost linear update time. In *Proc. of STOC '04*, pp. 184–191, 2004.
- [25] L. Roditty and U. Zwick. On dynamic shortest paths problems. In *Proc. of ESA '04*, Vol. 3221 of *LNCS*, pp. 580–591. Springer, 2004.