On Fully Dynamic Strongly Connected Components

Adam Karczmarz
University of Warsaw, Poland
IDEAS NCBR, Warsaw, Poland

Marcin Smulewicz
University of Warsaw, Poland

Abstract

We consider maintaining strongly connected components (SCCs) of a directed graph subject to edge insertions and deletions. For this problem, we show a randomized algebraic data structure with conditionally tight $O(n^{1.529})$ worst-case update time. The only previously described subquadratic update bound for this problem [Karczmarz, Mukherjee, and Sankowski, STOC'22] holds exclusively in the amortized sense.

For the less general dynamic strong connectivity problem, where one is only interested in maintaining whether the graph is strongly connected, we give an efficient deterministic black-box reduction to (arbitrary-pair) dynamic reachability. Consequently, for dynamic strong connectivity we match the best-known $O(n^{1.407})$ worst-case upper bound for dynamic reachability [van den Brand, Nanongkai, and Saranurak FOCS'19]. This is also conditionally optimal and improves upon the previous $O(n^{1.529})$ bound. Our reduction also yields the first fully dynamic algorithms for maintaining the minimum strong connectivity augmentation of a digraph.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases dynamic strongly connected components, dynamic strong connectivity, dynamic reachability

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.68

Funding *Adam Karczmarz*: Partially supported by the ERC CoG grant TUgbOAT no 772346 and the National Science Centre (NCN) grant no. 2022/47/D/ST6/02184.

1 Introduction

Two vertices of a directed graph $G = (V, E)$ are *strongly connected* if they can reach each other via a path in G. Pairwise strong connectivity is an equivalence relation and the *strongly connected components* of G are the equivalence classes of that relation. The graph G is called *strongly connected* if it has a single strongly connected component, or, equivalently, the transitive closure of G is a complete digraph. Testing strong connectivity and computing strongly connected components (SCCs) are among the most fundamental algorithmic problems on digraphs. Tarjan [24] famously showed a DFS-based linear-time algorithm for finding SCCs, which now, along with later alternatives [11, 23], is often taught in basic algorithms courses and appears in textbooks [9].

We study maintaining strong connectivity and strongly connected components in dynamic setting. A dynamic graph data structure is called *incremental* if it can handle edge insertions only, *decremental* if it can handle edge deletions only, and *fully dynamic* if it can handle both. When designing dynamic graph data structures, one is typically interested in optimizing both the (amortized or worst-case) update time of the data structure, and the query time. In the partially dynamic settings (i.e., incremental or decremental) one usually optimizes the total update time, i.e., the time needed to process the entire sequence of updates. In the following, let $n = |V|$ and $m = |E|$.

In this paper, we focus on the fully dynamic setting with single-edge updates. We are interested in maintaining the information about the *global* structure of SCCs of G, as opposed to supporting pairwise strong-connectivity queries. The information of interest may be either:

(a) a single bit indicating whether G is strongly connected (let us call this variant *SC*),
(b) the number of SCCs of G (*#SCC*),
(c) a representation of the SCCs allowing very efficient access to the individual SCCs, that is, computing the size in $O(\text{polylog}\, n)$ time, or reporting the elements in $O(\text{polylog}\, n)$ time per element (*SCCs*).

In particular, (c) can be achieved by maintaining an explicit $\Theta(n)$-sized partition of V into SCCs, which also allows for $O(1)$-time pairwise strong connectivity queries. On the other hand, using pairwise strong connectivity queries cannot easily provide any of the considered global information. In the following, for simplicity, we assume the desired information is explicitly recomputed after each update so that there is no need to consider query time. Thus, our sole goal is to optimize the *worst-case* update time.

Lower bounds. Note that if one is required to maintain the SCCs of a fully dynamic graph explicitly, then a single update can cause quite dramatic $\Omega(n)$-sized change in the set of SCCs (consider a directed cycle). Abboud and Vassilevska Williams [1] showed that even for maintaining a single-bit information whether G has more than two SCCs (the *SC2* problem), a data structure with $O(n^{1-\epsilon})$ amortized time[1] is unlikely, as it would break the Orthogonal Vectors conjecture, which is implied by SETH [15, 27]. The SETH-hardness of [1] does not seem to extend to *SC* though, which suggests that #SCC (or even SC2) might be a computationally harder problem.

The more recent OMv conjecture [14] implies that one cannot achieve $O(n^{1-\epsilon})$ update time even for SC. This suggests that for sparse graphs with $m = \widetilde{O}(n)$ the trivial recompute-from-scratch approach might be the best possible approach for SC and SCCs.

van den Brand, Nanongkai, and Saranurak [26] developed a few extensions of the OMv conjecture and proved, based on those, that fully dynamic s,t-reachability (where $s,t \in V$ are fixed) currently requires $\Omega(n^{1.406})$ amortized update time, and fully dynamic single-source reachability (with only the source s fixed) currently requires $\Omega(n^{1.528})$ time per update.[2] This means that the state-of-the-art data structures [26, 20] for the respective variants are near-optimal. By known reductions (see, e.g., [1]), one obtains that SC requires $\Omega(n^{1.406})$ update time, whereas SCCs require $\Omega(n^{1.528})$ update time.

Upper bounds. Abboud and Vassilevska Williams [1] showed that in order to have $\widetilde{O}(n^{2-\epsilon})$ amortized update time for SC using fast matrix multiplication (FMM) is essential, thus ruling out non-trivial combinatorial approaches. And indeed, FMM-based algebraic dynamic reachability algorithms [20, 26] allow breaking the quadratic update bound. This is because

[1] Abboud and Vassilevska Williams[1] actually write that $\Omega(m^{1-o(1)})$ is the best bound one can hope for. However, since this bound is a function of m exclusively, it is required to hold only for *some* density regime, e.g., sparse graphs.

[2] More specifically, based on their OMv conjecture variants, van den Brand, Nanongkai, and Saranurak [26] proved, for any $\epsilon > 0$, an $\Omega(\min_{a,b\in[0,1]}(n^{a+b} + n^{\omega(1,a,1)-a} + n^{\omega(1,a,b)-b}) \cdot n^{-\epsilon})$ lower bound for dynamic s,t-reachability, and an $\Omega(n^{1+\rho-\epsilon})$ lower bound for dynamic single-source reachability (SSR), where $\rho < 0.529$ satisfies $1 + 2\rho = \omega(1,\rho,1)$. Here $O(n^{\omega(a,b,c)})$ denotes the time needed to multiply an $n^a \times n^b$ boolean matrix by an $n^b \times n^c$ boolean matrix. These two conditional lower bounds become $\Omega(n^{1.406})$ and $\Omega(n^{1.528})$, respectively, assuming the current exponents of rectangular matrix multiplication. Furthermore, they become $\Omega(n^{1+1/4-\epsilon})$ and $\Omega(n^{1+1/2-\epsilon})$, respectively, assuming $\omega = 2$.

SC easily reduces to maintaining reachability to and from an arbitrary single vertex [26]. As a result, $O(n^{1.529})$ worst-case update time is possible in this case. This state-of-the-art bound does not match the $O(n^{1.406})$ lower bound [26], though. More generally, since the fully dynamic data structure of Sankowski [20] allows relatively cheap $O(n^{0.529})$-time arbitrary-pair reachability queries[3] at the cost of $O(n^{1.529})$ update time, this approach generalizes to maintaining some *at most* k distinct SCCs of G in $O(kn^{1.529})$ worst-case time per update. In particular, for $k = 2$ this trivially yields a solution to SC2 with $O(n^{1.529})$ worst-case update time. This matches the best known upper bound for SC. Consequently, the state-of-the-art bound for SC does not reflect the intuition of [1] that SC2 might be a harder problem.

The described approach, however, does not easily allow recomputing *all* SCCs of G, or even counting them, within subquadratic update time. In fact, the more general #SCC and SCCs problems do not seem to reduce to inspecting some $O(n^{2-\epsilon})$-sized subset of the reachability matrix of G. Nevertheless, Karczmarz, Mukherjee, and Sankowski [17] recently showed that SCCs (and thus #SCC) can be maintained in $O(n^{1.529})$ *amortized* time per update. Their approach is a mix of combinatorial and algebraic methods and critically depends on the efficiency of a *decremental SCCs* data structure. Bernstein, Probst Gutenberg, and Wulff-Nilsen [7] gave a near-optimal data structure maintaining SCCs *explicitly* under edge deletions with $\widetilde{O}(m)$ total update time. Although the $O(n^{1.529})$ amortized bound is near-optimal, the worst-case update time of the data structure of [7] might be $\Theta(n^2)$, and thus the worst-case update time of the fully dynamic SCCs data structure [17] may be as much as $\Theta(n^2)$ as well. To the best of our knowledge, no non-trivial worst-case bound for either #SCC or SCCs under fully dynamic edge updates has been described to date.

1.1 Our results

First of all, we close the gaps between the best known lower- and upper (worst-case) bounds for both fully dynamic strong connectivity (SC) and fully dynamic strongly connected components (SCC) in general directed graphs. See also Table 1. We achieve that by showing novel combinatorial reductions to the known fully dynamic arbitrary-pair reachability data structures [26, 20].

Fully dynamic strongly connected components. For fully dynamic SCCs we prove:

▶ **Theorem 1.** *Let G be a digraph. Suppose there is a fully dynamic reachability data structure $\mathcal{D}$ processing single-edge updates and arbitrary-pair queries on G in $U(n)$ and $Q(n)$ time respectively. Suppose the answers produced by $\mathcal{D}$ are correct with high probability[4].*

Then one can explicitly maintain the SCCs of G subject to single-edge insertions and deletions in $\widetilde{O}(U(n) + n \cdot Q(n))$ time per update (worst-case if the bounds $U(n), Q(n)$ are worst-case). The obtained data structure is Monte Carlo randomized. The answers produced are correct with high probability.

It is worth noting that the data structure trivially works against an adaptive adversary[5] since the revealed information, the SCCs, depend only on the current graph G. As far as the adversarial model of the assumed data structure $\mathcal{D}$ is concerned, no special requirements

[3] Throughout, we use the term *arbitrary-pair reachability* to refer to the situation when the endpoints s, t of a reachability pair of interest are a part of a query and may vary for different queries. The output of a query is a single bit indicating whether s can reach t in the (current) graph.

[4] That is, with probability at least $1 - 1/n^c$, where the constant $c \geq 1$ can be set arbitrarily.

[5] That can only see the output of the data structure, and not the random bits used by the data structure internally.

beyond high-probability correctness are needed. Indeed, the query interface of $\mathcal{D}$ does not leak any information about the potential random choices made by $\mathcal{D}$ internally (unless $\mathcal{D}$ errs, which we treat as a low-probability failure anyway) since the correct answers are uniquely determined by the graph maintained in $\mathcal{D}$.

Sankowski [20] showed a Monte Carlo randomized data structure with $U(n) = O(n^{1+\rho})$ and $Q(n) = O(n^{\rho})$ (both worst-case), where $\rho < 0.529$ is such that $1 + 2\rho = \omega(1, \rho, 1)$. As a result, Theorem 1 implies a data structure with $O(n^{1.529})$ worst-case update bound for fully dynamic SCCs. To the best of our knowledge, we are the first to achieve a non-trivial $O(n^{2-\epsilon})$ *worst-case* update time for SCCs and even #SCC. Moreover, via a folklore[6] reduction from fully dynamic single-source reachability and the conditional lower bound of [26], the obtained $O(n^{1.529})$ bound is tight for SCCs. Hence, our data structure constitutes the final answer to the problem (up to polylog factors), unless the *Mv-hinted Mv* variant of the OMv conjecture of [26, Conjecture 5.7] fails. That being said, it is not clear whether our bound is tight for the counting variant #SCC whose output is a single number, as opposed to n bits in SSR.

The worst-case update time guarantee allows applying Theorem 1 in the fault-tolerant model, where the goal is to recompute a graph property for a query set of at most k failed edges or vertices. Georgiadis, Italiano, and Parotsidis [12] showed that after optimal linear preprocessing, one can compute the SCCs of G after removing any single vertex/edge in optimal $O(n)$ time. Baswana, Choudhary, and Roditty [2] showed that after polynomial preprocessing, one can find the SCCs of G after removing k vertices/edges in $\widetilde{O}(2^k \cdot n)$ time. Thus, the known results could only handle at most $\log n$ failures faster than recompute-from-scratch, for any density of G. Our result implies that for $m = \Omega(n^{1.53})$ one can recompute SCCs faster than from scratch even under *polynomially* many (that is, $k = O(m/n^{1.529})$) vertex/edge failures[7]. That being said, the purely combinatorial data structures [2, 12] are significantly faster if, e.g., k is constant.

It is worth noting that a reduction of computing SCCs to a reachability problem has been known in the parallel setting [21]. That static reduction does not seem to be applicable in the dynamic setting using algebraic techniques, though. More specifically, Schudy [21] reduces computing SCCs of G to a number of adaptively generated instances of *multi-source reachability* (which in turn reduces to the single-source case by adding a super-source) on induced subgraphs of G of total size $\widetilde{O}(m)$. In the dynamic setting, spending time linear in m per update is prohibitive, and furthermore it is not clear how to efficiently simulate adaptively generated multi-source reachability queries for the following reasons. First, using the super-source trick would either require performing possibly $\Omega(n)$ edge updates (which are very costly, i.e., super-linear using algebraic methods) in the reachability data structure. If we wanted to avoid that, i.e., handle each source separately, we would possibly end up performing $\Omega(n^2)$ single-pair reachability queries. Our reduction for dynamic SCCs relies on being able to query $\widetilde{O}(n)$ arbitrary reachability pairs after each update. Crucially, these pairs cannot be grouped into, e.g., polylog(n) multi-source queries. Such an n-arbitrary-pairs reachability problem looks much more difficult even in the static setting and does not seem to be solvable in linear time like multi-source reachability. In fact, we believe no static bound beyond $O(\min(nm, n^{\omega}))$ is known.

---

[6] It is enough to maintain a graph G' obtained from G by adding a super-sink t and edges vt for all $v \in V$. Then for a query source s, vertices reachable from s in G are those in the SCC of s in $G' + ts$.

[7] Vertex failures can be easily simulated by splitting each $v \in V$ into $v_{\text{in}}, v_{\text{out}}$, that inherit the incoming and outgoing edges of v, resp, and introducing an edge $e_v = v_{\text{in}}v_{\text{out}}$. Then the failure of v is equivalent to the failure of e_v.

Strong connectivity. For the less general dynamic strong connectivity (SC), we give an even more efficient and *deterministic* black-box reduction to dynamic arbitrary-pair reachability.

▶ **Theorem 2.** *Let G be a digraph. Suppose there is a fully dynamic reachability data structure processing single-edge updates and arbitrary-pair reachability queries on G in at most $T(n)$ time. Then one can maintain whether G is strongly connected subject to single-edge insertions and deletions in $O(T(n)\log n)$ time per update (worst-case if the $T(n)$ bound is worst-case).*

van den Brand, Nanongkai, and Saranurak [26] showed a Monte Carlo randomized fully dynamic arbitrary-pair reachability data structure with $O(n^{1.407})$ worst-case update and query time. Therefore, Theorem 2 implies $O(n^{1.407})$ worst-case update bound for fully dynamic strong connectivity. This improves upon the known $O(n^{1.529})$ bound following from the trivial reduction to dynamic single-source reachability. By the conditional lower bound of [26], we essentially close the complexity of fully dynamic strong connectivity (up to polylogarithmic factors), unless the uMv-hinted uMv conjecture of [26, Conjecture 5.12] fails.

Whereas the obtained $O(n^{1.407})$ update bound is Monte Carlo randomized, this is only caused by the Schwartz-Zippel-lemma-style randomization inside the used reachability data structure. Obtaining a deterministic subquadratic fully dynamic reachability data structure would immediately imply a deterministic subquadaratic bound for SC.

Interestingly, our reduction *does not* seem to generalize to maintaining whether G has more than two SCCs (SC2), which was the case for the trivial reduction. As a result, $O(n^{1.529})$ currently remains the sharpest update bound for SC2. At the same time the tightest conditional lower bound we have for SC2 is $\Omega(n^{1.407})$ because SC2 does not seem to be easily reducible from dynamic single-source reachability, which was the case for dynamic SCCs. This is a consequence of the fact that the output in the SCC problem is n numbers, whereas in SC2 it is just a single bit. Obtaining a faster dynamic algorithm for SC2 or coming up with a plausible lower bound beyond $O(n^{1.407})$ is an interesting next step.

Table 1 Comparison of state-of-the art conditional lower- and upper bounds for the problems SC, SC2, #SCC, and SCC, and new upper bounds obtained via our reductions. Each problem in the table generalizes the preceding one. For SC and SCC, our obtained upper bounds are tight.

<table>
<tr><th>problem</th><th>known conditional lower bound</th><th>prev. worst-case update bound</th><th>previous amortized update bound</th><th>our new worst-case update bound</th></tr>
<tr><td>SC</td><td rowspan="3">$\Omega(n^{1.406})$
[1]+[26]</td><td rowspan="2">$O(n^{1.529})$
[26, 20]</td><td rowspan="2">same as worst-case</td><td>$O(n^{1.407})$
Theorem 2+[26]</td></tr>
<tr><td>SC2</td><td>–</td></tr>
<tr><td>#SCC</td><td rowspan="2">$O(n^2)$
trivial</td><td rowspan="2">$O(n^{1.529})$
[17]</td><td rowspan="2">$O(n^{1.529})$
Theorem 1+[20]</td></tr>
<tr><td>SCC</td><td>$\Omega(n^{1.528})$
[26] via SSR</td></tr>
</table>

Generalizations. The reduction of Theorem 2 does generalize in two different ways. First, within the same update bound (see Theorem 11) we can actually maintain the exact size $\chi(G)$ of the *minimum strong connectivity augmentation* [10] of G. The minimum strong connectivity augmentation is a smallest possible set of edges that one needs to add to G to make it strongly connected. $\chi(G)$ can be seen as an alternative and "orthogonal" (to the number of SCCs) measure of how much G *is not* strongly connected. As shown by Eswaran and Tarjan [10], in the static setting, the value $\chi(G)$ and some minimum strong connectivity augmentation can be computed in linear time.

As is typical for algebraic data structures[8], maintaining an *object* – some minimum strong augmentation in our case – is trickier than maintaining only the optimal *value* $\chi(G)$. Our algorithm can be nevertheless extended to maintain some $\chi(G)$-sized augmentation Y within the same $O(n^{1.407})$ worst-case bound and against an adaptive adversary, as long as $\chi(G) = O(n^{0.703})$. On the other hand, for larger $\chi(G)$, e.g., $\chi(G) = \Theta(n)$, some minimum augmentation Y can be maintained in $O(n^{1.529})$ worst-case time per update.

Interestingly, the weighted version of the minimum strong connectivity augmentation problem, where the costs of different potential edges to be added vary, is NP-hard [10].

Moreover, the reduction behind Theorem 2 enables *universal* reachability queries, even if G is not strongly connected. That is, in $O(n^{1.407})$ time, we can decide whether a query vertex v can reach all vertices in G, or find some vertex that v cannot reach. See Theorem 14. The $O(n^{1.529})$ query time follows trivially (simply issue all the n possible reachability queries from v) from the fully dynamic single-source reachability data structure [20].

1.2 Further related work

Since the known lower bounds [1, 14] imply that fully dynamic (or partially dynamic with worst-case bounds) SCCs data structures are possible only for sufficiently dense graphs and using algebraic methods, most of the previous work regarding maintaining SCCs dealt with partially dynamic settings with the goal of optimizing total update time.

Maintaining SCCs has been studied in the incremental setting [5, 3, 13]. The current best known randomized solution for sparse graphs [5] has $\widetilde{O}(m^{4/3})$ total update time, whereas the best known deterministic data structure [3] has $O(m^{3/2})$ total update time. For denser graphs, Bender et al. [3] gave a near-optimal deterministic data structure with $\widetilde{O}(n^2)$ total update time. All these data structures maintain SCCs explicitly.

Decremental SCCs maintenance has been studied even more extensively [6, 7, 8, 18, 19]. Bernstein, Probst Gutenberg, and Wulff-Nilsen [7] gave a randomized data structure with near-optimal $\widetilde{O}(m)$ total update time. The current best known deterministic total update time bound of $\widetilde{O}(mn^{2/3})$ for general digraphs is due to Bernstein, Probst Gutenberg, and Saranurak [6]. For planar graphs, the near-optimal $\widetilde{O}(n)$ total update time can be achieved deterministically [16].

In the case of (global) strong connectivity, incremental graph search from/to an arbitrary single source yields optimal $O(m)$ total update time. It is not clear if decremental strong connectivity is easier, in any sense, than decremental SCCs, and no specialized data structures beyond those for SCCs maintenance [7, 6] are known for decremental strong connectivity.

2 Preliminaries

In this paper we deal with *directed* graphs. We write $V(G)$ and $E(G)$ to denote the sets of vertices and edges of G, respectively. We omit G when the graph in consideration is clear from the context. A graph H is a *subgraph* of G, which we denote by $H \subseteq G$, if and only if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We write $e = uv \in E(G)$ when referring to edges of G. By G^{R} we denote G with edges reversed.

A sequence of vertices $P = v_1 \dots v_k$, where $k \geq 1$ is called an $s \to t$ path in G if $s = v_1$, $v_k = t$ and there is an edge $v_i v_{i+1}$ in G for each $i = 1, \dots, k-1$. We sometimes view a path P as a subgraph of G with vertices $\{v_1, \dots, v_k\}$ and (possibly zero) edges $\{v_1 v_2, \dots, v_{k-1} v_k\}$.

[8] For example, for dynamic reachability and exact distances, the known algebraic data structures [25, 26, 20] are polynomially faster than the known solutions to their respective path-reporting variants [4, 17].

For convenience, we sometimes consider a single edge uv a path. If P_1 is a $u \to v$ path and P_2 is a $v \to w$ path, we denote by $P_1 \cdot P_2$ (or simply P_1P_2) a path obtained by concatenating P_1 with P_2. A vertex $t \in V(G)$ is *reachable* from $s \in V(G)$ if there is an $s \to t$ path in G.

3 Fully dynamic strongly connected components

In this section we describe our fully dynamic strongly connected components data structure.

Suppose there exists a reachability data structure maintaining an n-vertex digraph subject to single-edge insertions and deletions and answering arbitrary-pair reachability queries. Denote by $U(n)$ and $Q(n)$ the update and query (resp.) time bounds of the assumed data structure.

We start by proving the following key observation, that tracking the vertices $v \in V$ strongly connected to *some* vertex t out of a chosen subset $T \subseteq V$ can be reduced to maintaining some n cells of the transitive closure matrix (reachability pairs) of a certain auxiliary graph G_T.

▶ **Lemma 3.** *Let $G = (V, E)$ be a digraph, and let $T \subseteq V$. Let G' be a graph with vertices $V' = \{v' : v \in V\}$, and edges $E' = \{u'v' : uv \in E\}$, i.e., G' is a copy of G with vertices primed. Consider a graph G_T obtained from $G \cup G'$ by adding edges tt' for each $t \in T$. Then, for any $v \in V$, v is strongly connected with* some *$t \in T$ in G if and only if there is a path $v \to v'$ in G_T.*

Proof. Suppose that v is strongly connected with some $t \in T$ in G. Then, there exists a path $P = v \to t$ in G, and a path $Q = t \to v$ in G. As a result, there is a corresponding path $Q' = t' \to v'$ in G'. Since $tt' \in E(G_T)$, we conclude there is a $v \to v'$ path $P \cdot tt' \cdot Q'$ in G_T.

Now suppose there is a $v \to v'$ path in G_T. Note that a path R starting in the G-part of G_T can only depart from G for G' through an edge tt', $t \in T$, and it cannot go back to G. As a result, R can be expressed as $R_1 \cdot (tt') \cdot R'_2$, where $R_1 = v \to t$ is a path in G, and R'_2 is a $t' \to v'$ path in G'. But R'_2 has a corresponding path $R_2 = t \to v$ in G. The paths R_1, R_2 certify that v and t are strongly connected. Consequently, v is strongly connected to some vertex of T in G. ◀

To make use of Lemma 3 we will employ the *unique witness trick* (usually attributed to [22]) that have been used for computing witnesses of various matrix products, also in the dynamic setting [4]. However, in our case, we will use this technique with a conceptually very different purpose: to *select* a certain *root* of every SCC that ever appears in G throughout edge updates.

▶ **Lemma 4.** [22] *Let $X \subseteq V$ be a subset with $n/2^{k+1} \le |X| \le n/2^k$ for an integer $k \ge 0$. Let $S \subseteq V$ be a subset obtained by sampling 2^k elements of V uniformly at random and independently (with replacement). Then, with probability at least $1/6$, $|X \cap S| = 1$.*

Proof. The probability that only the i-th sampled vertex is a unique element of $X \cap S$ is

$$\frac{|X|}{n} \cdot \left(1 - \frac{|X|}{n}\right)^{2^k - 1} \ge \frac{1}{2^{k+1}} \cdot \left(1 - \frac{1}{2^k}\right)^{2^k - 1} \ge \frac{1}{2^{k+1}e}.$$

The probability that $|X \cap S| = 1$ is at least the probability of one of the above disjoint events happening, i.e., at least $2^k \cdot \frac{1}{2^{k+1}e} = \frac{1}{2e} \ge 1/6$. ◀

Let $\ell = (\gamma + 2) \cdot \log_{6/5} n$ for a constant $\gamma > 0$ that can be adjusted to control the error probability. Let B be the smallest integer such that $n < 2^B$, so that $B = O(\log n)$. For a non-negative integer m, put $[m] := \{0, \ldots, m\}$. Let us identify V with the B-bit numbers $\{1, \ldots, n\}$. Moreover, let

$$\mathcal{T} = \{T_{k,l,b} : k \in [B-1], l \in [\ell - 1], b \in [B-1]\}$$

be a family of subsets of V obtained as follows. For $k \in [B-1], l \in [\ell - 1]$ let $T_{k,l}$ be a subset of V obtained by sampling 2^k elements of V uniformly at random and independently, with replacement. Then, set $T_{k,l,b}$ to be a subset of numbers $v \in T_{k,l}$ such that the b-th (0-based) bit of v in the binary equals 1. Clearly, the family $\mathcal{T}$ has $O(\log^3 n)$ subsets.

First of all, we maintain the graph G itself in the assumed fully dynamic reachability data structure $\mathcal{D}$. For each $T \in \mathcal{T}$, we set up another assumed data structure $\mathcal{D}_T$ maintaining the graph G_T, as defined in Lemma 3. Note that each edge update to the graph G is reflected using just two edge updates to each graph G_T since G_T consists of two copies of G. As a result, every edge update can be processed in $O(U(n) \cdot |\mathcal{T}|) = \widetilde{O}(U(n))$ time (worst-case if the bound $U(n)$ is worst-case).

We now describe how the SCCs of G are recomputed after an update. We build an auxiliary graph H whose construction follows. For each $v \in V$, $k \in [B-1]$, and $l \in [\ell - 1]$, we do the following. For every $b \in [B-1]$, we query the data structure $D_{T_{k,l,b}}$ whether there exists a path $v \to v'$ in $G_{T_{k,l,b}}$. Let $s_{v,k,l}$ be a number (vertex) such that the b-th bit is set to 1 if and only if the corresponding path in $G_{T_{k,l,b}}$ exists. If $s_{v,k,l} \in \{1, \ldots, n\}$ (so that $s_{v,k,l}$ is indeed a vertex of G) and v *is strongly connected with* $s_{v,k,l}$ (which can be tested using two queries to $\mathcal{D}$), we add an undirected edge $\{v, s_{v,k,l}\}$ to H.

Observe that the graph H is constructed by performing $\widetilde{O}(n)$ queries to the assumed data structures $\mathcal{D}$ and $\mathcal{D}_T$, $T \in \mathcal{T}$. The cost of these queries is $\widetilde{O}(n \cdot Q(n))$ (worst-case if $Q(n)$ is worst-case).

Having constructed the undirected graph H, we output its connected components as the SCCs of G. Since H has $\widetilde{O}(n)$ edges, this can be done in $\widetilde{O}(n)$ additional worst-case time.

▶ **Lemma 5.** *With probability at least $1 - 1/n^\gamma$, the connected components of H and the strongly connected components of G are equal.*

Proof. By the construction, if two vertices $u, v \in V$ are connected by an edge (or more generally, a path) in H, then they are strongly connected in G. Hence, we only need to prove that if $u, v \in V$ are strongly connected in G, then they are connected in H as well.

Let C be the SCC of G containing both u and v. Let $z \in [B-1]$ be such an integer that we have $n/2^{z+1} \leq |C| \leq n/2^z$. By Lemma 4, for each $l \in [\ell - 1]$, $T_{z,l}$ satisfies $|T_{z,l} \cap C| = 1$ with probability at least $1/6$. As a result, with probability at least $1 - (5/6)^\ell \geq 1 - 1/n^{\gamma+2}$, at least one of the sets $T_{z,l}$, say $T_{z,y}$, satisfies $|T_{z,y} \cap C| = 1$. Let s be the unique element of $T_{z,y} \cap C$. Observe that for $b \in [B-1]$, we have $T_{z,y,b} \cap C = \{s\}$ if the b-th bit of s equals 1 and $T_{z,y,b} \cap C = \emptyset$ otherwise. As a result, by Lemma 3, for both u and v the query to the data structure $\mathcal{D}_{T_{z,y,b}}$ will return true if and only if the b-th bit of s is 1. Equivalently, we will have $s = s_{v,k,l} = s_{u,k,l} \neq 0$. Since both edges $\{v, s\}$ and $\{u, s\}$ are added to H, u and v are indeed connected in H.

As there are at most n^2 pairs (u, v), the implications hold with probability at least $1 - n^2 \cdot (1/n^{\gamma+2}) = 1 - 1/n^\gamma$. ◀

Note that the above lemma holds for any version of G since, unless the algorithm makes a mistake (which happens with low probability), we do not reveal any random bits to the adversary as the output is always unique. To summarize, we obtain the following.

▶ **Theorem 1.** *Let G be a digraph. Suppose there is a fully dynamic reachability data structure $\mathcal{D}$ processing single-edge updates and arbitrary-pair queries on G in $U(n)$ and $Q(n)$ time respectively. Suppose the answers produced by $\mathcal{D}$ are correct with high probability.*

Then one can explicitly maintain the SCCs of G subject to single-edge insertions and deletions in $\widetilde{O}(U(n) + n \cdot Q(n))$ time per update (worst-case if the bounds $U(n), Q(n)$ are worst-case). The obtained data structure is Monte Carlo randomized. The answers produced are correct with high probability.

Sankowski [20] showed that there exists a fully dynamic reachability data structure with $U(n) = \widetilde{O}(n^{1+\rho})$ worst-case update time and $Q(n) = \widetilde{O}(n^{\rho})$ query time, where $\rho < 0.529$ is such that $1 + 2\rho = \omega(1, \rho, 1)$. By Theorem 1 it follows that there exists a fully dynamic data structure maintaining SCCs explicitly with $\widetilde{O}(n^{1+\rho}) = O(n^{1.529})$ worst-case update time.

4 Reducing dynamic strong connectivity to dynamic reachability

In this section we show how to maintain whether the graph G is strongly connected subject to edge insertions and deletions using a fully dynamic reachability data structure supporting single-edge updates and arbitrary-pair reachability queries. Our reduction is deterministic and incurs only polylogarithmic overhead. Again, assume that $n < 2^B$ for an integer $B = O(\log n)$. The vertices are numbered 1 through n and thus can be seen as B-bit numbers.

Let us define a *source strongly connected component* of G to be an SCC of G with no incoming edges from other SCCs of G. Clearly, if G is strongly connected, it has precisely one source SCC. The main idea is to maintain a *source set* $\mathcal{Z} \subseteq V$ such that $\mathcal{Z}$ contains *precisely one* vertex z of every source SCC of G. For example, if G is strongly connected, then $\mathcal{Z}$ can be any single-element subset of V. We also have the following simple property.

▶ **Observation 6.** *For every vertex $v \in V(G)$ there exists some source SCC S of G such that v is reachable from every $s \in S$.*

Proof. Let C be the SCC of v in G. Let G^* be the *condensation* of G obtained from G by contracting the SCCs. Then, G^* is a DAG. In a DAG, every vertex is reachable from a source (that is, a vertex with no incoming edges). In particular, C is reachable from some source S of G^*. Consequently, any vertex $s \in S$ can reach v in G by construction of G^*. ◀

The following lemma shows how a data structure maintaining some source set of G can be used to maintain strong connectivity of G.

▶ **Lemma 7.** *Let G^R be the reverse of G. Let $\mathcal{Z}$ be some source set of G and let $\mathcal{Z}^R$ be some source set of G^R. Then G is strongly connected if and only if $|\mathcal{Z}| = |\mathcal{Z}^R| = 1$ and $t \in \mathcal{Z}^R$ can reach $s \in \mathcal{Z}$ in G.*

Proof. Clearly, if G is strongly connected, then $\mathcal{Z} = \{s\}$, $\mathcal{Z}^{\mathrm{R}} = \{t\}$ for some $s, t \in V$. Moreover, t can reach s in G.

Now suppose $\mathcal{Z} = \{s\}$ and $\mathcal{Z}^{\mathrm{R}} = \{t\}$. Consider any $x, y \in V$. By Observation 6, y is reachable from s. Similarly, by Observation 6 applied to G^{R}, x is reachable from t in G^{R}, i.e., t is reachable from x in G. By the assumption, t can reach s in G. Thus, there exist a path $x \to t \to s \to y$ in G. Since the pair x, y was arbitrary, G is indeed strongly connected. ◀

By Lemma 7, we can maintain strong connectivity using a fully dynamic source set data structure built on G, a fully dynamic source set data structure built on G^{R}, and a fully dynamic reachability data structure built on G. Every edge update translates to a single edge update in the three data structures, and a single query to the reachability data structure.

Let us now focus on maintaining a source set $\mathcal{Z}$ of G subject to edge updates. We have:

▶ **Lemma 8.** *For any $w \in V$, there exists at most one $z \in \mathcal{Z}$ such that w can reach z in G. Moreover, w is in a source SCC (containing z) if and only if such a $z \in \mathcal{Z}$ exists.*

Proof. Let $z \in \mathcal{Z}$ and let C be the source SCC of G with $z \in C$. By the definition of a source SCC, the only vertices of V that can reach z are those in C. As a result, if w can reach z, then $w \in C$. Therefore, $w \notin C'$ for other source SCCs $C' \neq C$, and thus w cannot reach any other $z' \in \mathcal{Z}$, $z' \neq z$. On the other hand, if $w \in C$ for some source SCC C, then w can clearly reach the unique element of $\mathcal{Z} \cap C$. ◀

Maintained data structures. First of all, let $\bar{G}$ be the graph G extended with a super source $s \notin V(G)$ and edges sz for all $z \in \mathcal{Z}$. We store $\bar{G}$ in a fully dynamic reachability data structure $\mathcal{D}$. Note that using a single reachability query to $\mathcal{D}$ we can check whether some query vertex $v \in V$ is reachable from $\mathcal{Z}$ in $\bar{G}$. Since $G \subseteq \bar{G}$ and s has no incoming edges, the data structure $\mathcal{D}$ can also handle usual reachability queries in G.

For $b \in [B-1]$, let G_b be the graph G extended with a super sink $t \notin V(G)$, and edges zt for all $z \in \mathcal{Z}$ such that the b-th bit of the number z equals 1. We store the $O(\log n)$ graphs G_b in fully dynamic reachability data structures $\mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$. Note that since every $v \in V$ has at least one bit set, $\bigcup_{i=0}^{B-1} G_b$ contains edges zt for all $z \in \mathcal{Z}$. As a result, a vertex $w \in V$ can reach some vertex of $\mathcal{Z}$ if and only if w can reach t in $\bigcup_{i=0}^{B-1} G_b$. Moreover, by Lemma 8, if w can reach some vertex of $\mathcal{Z}$, then it can reach precisely one such $z \in \mathcal{Z}$. The individual data structures G_b can be used to find that z: observe that w can reach t (equivalently, reach z) in G_b precisely if and only if z has the b-th bit set. As a result, we have the following.

▶ **Lemma 9.** *For any $w \in V$ one can find the unique $z \in \mathcal{Z}$ that w can reach in G (if one exists) using $O(\log n)$ queries to the fully dynamic reachability data structures $\mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$.*

Whenever the graph G is updated or the set $\mathcal{Z}$ undergoes an update, the update is passed to all the $O(\log n)$ relevant data structures $\mathcal{D}$ and $\mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$ so that they reflect the current graphs $\bar{G}$ and $G_0, \ldots, G_{B-1}$ respectively.

We now proceed with describing how the updates are processed. In the following, denote by G' the graph G after the edge update, whereas G denotes the graph before the edge update considered.

Edge insertions. First consider an insertion of edge uv. If there exists a path $u \to v$ in G (a single query to $\mathcal{D}$), the SCCs of G do not change due to the insertion. No source SCC gets an incoming edge from another SCC. Therefore, $\mathcal{Z}$ remains a valid source set.

Let us thus assume that there is no $u \to v$ path in G. In particular, u and v are not strongly connected in G. Let us denote by S_v the SCC containing v in G. Inserting uv cannot break any of the existing SCCs of G, but might cause some SCCs of G (in particular S_v) merge in G'. Let S^* denote the SCC of v in G'. We have $S_v \subseteq S^*$, but potentially $S^* = S_v$.

Let us now argue that S_v is the only SCC of G that might lose the status of a source SCC due to the insertion. Indeed, if $S_v \neq S^*$, then every other SCC $S' \neq S_v$ of G, such that $S' \subset S^*$, is reachable from S_v since the insertion of uv makes S_v and S' strongly connected. As a result, S' is not a source SCC of G. Moreover, the respective sets of incoming inter-SCC edges of other SCCs S'' of G such that $S'' \cap S^* = \emptyset$ do not change, so such S'' is a source SCC of G' if and only if S'' is a source SCC of G.

By Lemmas 8 and 9 we can test whether S_v is a source SCC (and possibly find the unique $z \in S_v \cap \mathcal{Z}$) by performing $O(\log n)$ queries to the data structures $\mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$. The next step is to remove z (if it exists) from $\mathcal{Z}$, and update all the data structures $\mathcal{D}, \mathcal{D}_0, \ldots, \mathcal{D}_b$ accordingly. Next, we apply the insertion of uv to all these data structures so that they store

(supergraphs) of the updated graph G'. Observe that at this point the set $\mathcal{Z}$ might miss at most one element, if S^* turns out to be a source SCCs of G'. In order to detect whether this is the case we check whether $v \in S^*$ can be reached from $\mathcal{Z}$ in G'. Recall that this amounts to a single query to $\mathcal{D}$ about the existence of a path from s to u in $\bar{G}$. If so, S^* is reachable from some other SCC, and thus is not a source SCC. Otherwise, S^* is indeed a source SCC and thus we add to $\mathcal{Z}$ an arbitrary element of S^*, e.g., the vertex v, so that $\mathcal{Z}$ again becomes a valid source set of G'.

Edge deletions. Edge deletions can be handled essentially symmetrically to edge insertions. Suppose an edge $uv \in E(G)$ is deleted. Again, if the deletion does not make v unreachable from u (which can be tested by performing a single update and query on $\mathcal{D}$), neither the SCCs of G nor the source SCCs of G change. So suppose u cannot reach v after the deletion.

Let S'_v be the SCC containing v in G' and let S^* be the SCC containing v in G. We have $S'_v \subseteq S^*$ and potentially $S'_v = S^*$. If S^* is a source SCC in G, we can find $z \in S^* \cap \mathcal{Z}$ using $O(\log n)$ reachability queries from v to t in $\mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$ by Lemmas 8 and 9.

Observe that S'_v is the only SCC contained in S^* that might potentially be a source SCC of G': if $S' \neq S'_v$ is an SCC of G', and $S' \subset S^*$, then S' is surely reachable from S'_v in G' and thus S' is not a source SCC of G'. Similarly as we did when processing an insertion, we remove z (if exists) from $\mathcal{Z}$. At this point, $\mathcal{Z}$ might need inserting at most one element in order to become a valid source set of G'. This is the case precisely when S'_v is a source SCC of G'. We can check that, again, by issuing a single $s \to v$ reachability query to $\mathcal{D}$ after the deletion is reflected in $\mathcal{D}$. If S'_v happens to be a source SCC of G', we add $v \in S'_v$ to $\mathcal{Z}$.

Note that processing an edge update causes at most two element updates to the set $\mathcal{Z}$. Each of these updates, and also the edge update itself, is reflected in $O(\log n)$ fully dynamic reachability data structures $\mathcal{D}, \mathcal{D}_0, \ldots, \mathcal{D}_{B-1}$. Moreover, only $O(\log n)$ reachability queries are performed on these data structures. Consequently, we obtain the following.

▶ **Lemma 10.** *Suppose there is a fully dynamic reachability data structure processing single-edge updates and arbitrary-pair reachability queries on G in at most $T(n)$ time. Then one can maintain a source set $\mathcal{Z}$ of G explicitly subject to single-edge insertions and deletions in $O(T(n) \log n)$ time per update (worst-case if the $T(n)$ bound is worst-case).*

By Lemma 10 and the discussion after Lemma 7, we obtain:

▶ **Theorem 2.** *Let G be a digraph. Suppose there is a fully dynamic reachability data structure processing single-edge updates and arbitrary-pair reachability queries on G in at most $T(n)$ time. Then one can maintain whether G is strongly connected subject to single-edge insertions and deletions in $O(T(n) \log n)$ time per update (worst-case if the $T(n)$ bound is worst-case).*

The currently best known bound on the maximum of query time and update time of a fully dynamic arbitrary-pair reachability data structure is $T(n) = O(n^{1.407})$ worst-case (Monte Carlo randomized) due to [26]. Consequently, Theorem 2 combined with [26] yields $O(n^{1.407})$ worst-case update time for fully dynamic strong connectivity.

Strong connectivity augmentation. Due to a result of Eswaran and Tarjan [10], using Lemma 10 we can actually prove a more general result.

▶ **Theorem 11.** *Let G be a digraph. Let $T(n)$ be defined as in Theorem 2. Then one can maintain the size $\chi(G)$ of a minimum strong connectivity augmentation of G subject to single-edge insertions and deletions in $O(T(n) \log n)$ time per update. (worst-case if the $T(n)$ bound is worst-case).*

Proof. Let $\mathcal{Z}$ and $\mathcal{Z}^{\mathrm{R}}$ be source sets of G and G^{R} respectively. Define a *sink SCC* to be an SCC of G that is a source SCC in G^{R}. Eswaran and Tarjan [10] prove that if G is not strongly connected, then $\chi(G)$ equals $\max(s,t)+q$, where s denotes the number of source SCCs that are not sink SCCs, t denotes the number of sink SCCs that are not source SCCs, and q denotes the number of "isolated SCCs" that are both source and sink SCCs. If G is strongly connected then $\chi(G)=0$ holds trivially.

Note that with s,t,q defined like this, we have $|\mathcal{Z}|=s+q$ and $|\mathcal{Z}^{\mathrm{R}}|=t+q$. Hence, if G is not strongly connected, then $\max(|\mathcal{Z}|,|\mathcal{Z}^{\mathrm{R}}|)=\max(s,t)+q=\chi(G)$. As a result, it is enough to maintain some sets $\mathcal{Z}$ and $\mathcal{Z}^{\mathrm{R}}$, and test whether G is strongly connected using Lemmas 10 and 7. Clearly, we need only $O(T(n)\log n)$ time for this. ◀

Again, by combining Theorem 11 with [26], we obtain that the size $\chi(G)$ of a minimum strong connectivity augmentation can be maintained in $O(n^{1.407})$ worst-case time per update.

Theorem 11 does not immediately yield any actual set Y of edges such that $|Y|=\chi(G)$ and $G+Y$ is strongly connected. We now discuss how such an augmentation Y can be maintained. To this end we now sketch the method [10] of obtaining an augmentation with $\max(|\mathcal{Z}|,|\mathcal{Z}^R|)$ edges in case G is not strongly connected. Observe that in such a case, the quantity $\max(|\mathcal{Z}|,|\mathcal{Z}^R|)$ is clearly a lower bound on the size of Y.

Suppose wlog. that $|\mathcal{Z}|\geq|\mathcal{Z}^R|$. For each $z\in\mathcal{Z}$, let $t_z\in\mathcal{Z}^R$ be arbitrary such that z can reach t_z in G; note that at least one such t_z always exists. Similarly, for each $z'\in\mathcal{Z}^R$, let $s_{z'}\in\mathcal{Z}$ be arbitrary such that z' is reachable from $s_{z'}$ in G.

Consider a set of edges $F=\{zt_z : z\in\mathcal{Z}\}\cup\{s_{z'}z' : z'\in\mathcal{Z}^R\}$. Let M be a maximal matching in F (possibly allowing self-loops), that is, a maximal subset $\{y_1y'_1,\ldots,y_ky'_k\}\subseteq F$ such that all $y_1,\ldots,y_k$ are distinct and all $y'_1,\ldots,y'_k$ are distinct.

Put $A:=\{y_1,\ldots,y_k\}$, and $B=\{y'_1,\ldots,y'_k\}$. Moreover, put $\mathcal{Z}\setminus A=\{z_1,\ldots,z_p\}$ and $\mathcal{Z}'\setminus B=\{z'_1,\ldots,z'_q\}$, where $p\geq q$. By the maximality of M, we have that for all $z\in\mathcal{Z}\setminus A$, $t_z\in B$, that is, z can reach some vertex from B in G. Similarly, for all $z'\in\mathcal{Z}^R\setminus B$, z' is reachable from a vertex of A.

Put $y_{k+1}:=y_1$. Consider an augmentation:

$$Y=\{y'_iy_{i+1} : i=1,\ldots,k\}\cup\{z'_{\max(i,q)}z_i : i=1,\ldots,p\}.$$

We have $|Y|=k+p=|\mathcal{Z}|=\max(|\mathcal{Z}|,|\mathcal{Z}^R|)$. Eswaran and Tarjan [10] argue rather easily that $G+Y$ is strongly connected: (1) the source and sink vertices in $A\cup B$ are pairwise strongly connected since they are arranged in a cycle, (2) every vertex in $\mathcal{Z}^R\setminus B$ can reach $A\cup B$ via $\mathcal{Z}\setminus A$ and the edges in Y, and (3) every vertex in $\mathcal{Z}\setminus A$ can be reached from $A\cup B$ via $\mathcal{Z}^R\setminus B$ and the edges in Y.

The above construction reduces, in $O(n)$ time, the problem of maintaining an optimal Y to maintaining:

(1) for each vertex $z\in\mathcal{Z}$, an arbitrary vertex $t_z\in\mathcal{Z}^R$ reachable from z in G,
(2) for each vertex $z'\in\mathcal{Z}^R$, an arbitrary vertex $s_{z'}\in\mathcal{Z}$ that can reach z' in G.

If $\chi(G)=\max(|\mathcal{Z}|,|\mathcal{Z}^R|)$ is small enough, we can achieve the above by simply maintaining the $(\mathcal{Z}\cup\mathcal{Z}^R)\times(\mathcal{Z}\cup\mathcal{Z}^R)$ submatrix of the transitive closure of G. van den Brand, Forster, and Nazari [25] showed the following.

▶ **Theorem 12.** [25] *Let G be a directed graph. Let S be a dynamic subset of V be such that $|S|=O(n^{0.85})$ at all times. There exists a data structure explicitly maintaining reachability between all-pairs of vertices in S subject to fully dynamic single-edge updates to G and single-element additions/deletions to S in $O(n^{1.407}+|S|^2)$ worst-case time per update.*

Recall that if G is subject to fully dynamic single-edge updates, then $\mathcal{Z}$ and $\mathcal{Z}^R$ undergo at most two element updates per update to G. As a result, by putting $S = \mathcal{Z} \cup \mathcal{Z}^R$ in the above theorem and combining with Lemma 10, we can maintain a minimum strong connectivity augmentation Y in $O(n^{1.407})$ time as long as $\chi(G)^2 = O(n^{1.407})$.

If $\chi(G) = \Theta(n)$, however, the above method would have quadratic update time. We now argue that if we expect $\chi(G)$ to be large, then some desired sets $\{t_z \in \mathcal{Z}^R : z \in \mathcal{Z}\}$, and $\{s_{z'} \in \mathcal{Z} : z' \in \mathcal{Z}^R\}$ can be maintained in $O(n^{1.529})$ worst-case time per update anyway. This is possible using the below simple *existential reachability* data structure.

▶ **Lemma 13.** *Let G be a directed graph. Suppose there is a fully dynamic reachability data structure processing single-edge updates and arbitrary-pair reachability queries on G in $U(n) = \mathrm{poly}(n)$ and $Q(n) = \mathrm{poly}(n)$ time respectively.*

Let $X \subseteq V$ be a dynamic subset. There is a data structure maintaining G subject to fully dynamic edge updates and X subject to single-element insertions and deletions with $\widetilde{O}(U(n))$ update time supporting the following queries in $\widetilde{O}(Q(n))$-time. Given a query vertex $s \in V$, find some vertex $x \in X$ that s can reach in G (if such an x exists).

Proof sketch. Suppose wlog. that the vertices of G are identified with $\{1, \ldots, n\}$ and n is a power of two. Let G' be obtained from G as follows. First, we augment G with an auxiliary full binary tree $\mathcal{T}$ with n leaves labeled $l_1, \ldots, l_n$ from left to right, so that $V(\mathcal{T}) \cap V(G) = \emptyset$. The edges in $\mathcal{T}$ are directed from children to their parents. Second, for every $x \in X$, we add an edge xl_x to G'. Note that the graph G' has at most $3n$ vertices.

We maintain the graph G' using the assumed reachability data structure $\mathcal{D}$. Whenever G is subject to an edge insertion or deletion, that update is also applied to G'. Similarly, if an element x is inserted/deleted from X, this is reflected in G' by adding/deleting the edge xl_x.

The binary tree $\mathcal{T}$ allows locating some (or even the minimum-label) vertex of X that a query vertex s can reach within $O(\log n)$ queries to $\mathcal{D}$. Indeed, note that s can reach a vertex $w \in V(\mathcal{T})$ in G' if and only if s can reach in G some vertex $x \in X$ such that w is an ancestor of the leaf $l_x \in V(\mathcal{T})$. In particular, s can reach X in G if and only if s can reach the root of $\mathcal{T}$ in G'. If this is the case, the search starts in the root of $\mathcal{T}$ and descends down the tree maintaining the invariant that one of the leaves in the current subtree is reachable from s in G'. Once we reach a leaf l_x, we report $x \in X$ as reachable from s. ◀

With Lemma 13 in hand (applied twice, to the graph G with $X := \mathcal{Z}^R$ and to the graph G^R with set $X := \mathcal{Z}$), using fully dynamic reachability data structure of [20] with $O(n^{1.529})$ worst-case update time and $O(n^{0.529})$ query time, we can find the desired sets by issuing $O(n)$ existential reachability queries. Recall that the sets $\mathcal{Z}$ and $\mathcal{Z}^R$ undergo $O(1)$ updates per edge update to G, so the worst-case update time is $O(n^{1.529})$.

Finally, we note that the described algorithms for maintaining the minimum strong connectivity augmentation Y work against an adaptive adversary. Indeed, they are deterministic if provided with the required reachability information about G. That in turn is uniquely defined and maintained correctly with high probability by the data structures of [25, 20].

Universal reachability. By maintaining the source set $\mathcal{Z}$, we can also achieve the following.

▶ **Theorem 14.** *Let G be a digraph. Let $T(n)$ be defined as in Theorem 2. There exists a data structure maintaining G under single-edge insertions and deletions and answering queries whether a given vertex v can reach all vertices in G (and if not, providing an example of an unreachable vertex) with $\widetilde{O}(T(n))$ update and query time.*

Proof. Let $\mathcal{Z}$ be again a source set of G. Clearly, if $|\mathcal{Z}| > 1$, then no vertex of G can reach all other vertices. Otherwise, if $|\mathcal{Z}| = 1$, then some v can reach all of V if and only if v can reach (or, equivalently, is strongly connected to) the only element $z \in \mathcal{Z}$. As a result, given $\mathcal{Z}$ and the data structure $\mathcal{D}$, we can check whether v can reach all vertices in G by issuing a $v \to z$ query to $\mathcal{D}$ in $\widetilde{O}(T(n))$ time. If v cannot reach some vertex of G, then it cannot reach some $z' \in \mathcal{Z}$. Since there is at most one $z \in \mathcal{Z}$ that v can reach, to find some unreachable $z' \in \mathcal{Z}$, it is enough to issue two queries to $\mathcal{D}$. ◀

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 434–443. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.53.

2 Surender Baswana, Keerti Choudhary, and Liam Roditty. An efficient strongly connected components algorithm in the fault tolerant model. *Algorithmica*, 81(3):967–985, 2019. doi:10.1007/s00453-018-0452-3.

3 Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Robert E. Tarjan. A new approach to incremental cycle detection and related problems. *ACM Trans. Algorithms*, 12(2):14:1–14:22, 2016. doi:10.1145/2756553.

4 Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1836–1855. SIAM, 2021. doi:10.1137/1.9781611976465.110.

5 Aaron Bernstein, Aditi Dudeja, and Seth Pettie. Incremental SCC maintenance in sparse graphs. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 14:1–14:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.14.

6 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1123–1134. IEEE, 2020. doi:10.1109/FOCS46700.2020.00108.

7 Aaron Bernstein, Maximilian Probst, and Christian Wulff-Nilsen. Decremental strongly-connected components and single-source reachability in near-linear time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 365–376. ACM, 2019. doi:10.1145/3313276.3316335.

8 Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Jakub Łącki, and Nikos Parotsidis. Decremental single-source reachability and strongly connected components in $\tilde{o}(m\sqrt{n})$ total update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 315–324. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.42.

9 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

10 Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976. doi:10.1137/0205044.

11 Harold N. Gabow. Path-based depth-first search for strong and biconnected components. *Inf. Process. Lett.*, 74(3-4):107–114, 2000. doi:10.1016/S0020-0190(00)00051-X.

12 Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. *SIAM J. Comput.*, 49(5):865–926, 2020. doi:10.1137/19M1258530.

13 Bernhard Haeupler, Telikepalli Kavitha, Rogers Mathew, Siddhartha Sen, and Robert Endre Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Trans. Algorithms*, 8(1):3:1–3:33, 2012. doi:10.1145/2071379.2071382.

14 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 21–30. ACM, 2015. doi:10.1145/2746539.2746609.

15 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

16 Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, and Piotr Sankowski. Decremental single-source reachability in planar digraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1108–1121. ACM, 2017. doi:10.1145/3055399.3055480.

17 Adam Karczmarz, Anish Mukherjee, and Piotr Sankowski. Subquadratic dynamic path reporting in directed graphs against an adaptive adversary. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1643–1656. ACM, 2022. doi:10.1145/3519935.3520058.

18 Jakub Łącki. Improved deterministic algorithms for decremental reachability and strongly connected components. *ACM Trans. Algorithms*, 9(3):27:1–27:15, 2013. doi:10.1145/2483699.2483707.

19 Liam Roditty and Uri Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM J. Comput.*, 37(5):1455–1471, 2008. doi:10.1137/060650271.

20 Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *45th Symposium on Foundations of Computer Science FOCS 2004*, pages 509–517. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.25.

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

22 Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995. doi:10.1006/jcss.1995.1078.

23 M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981. doi:10.1016/0898-1221(81)90008-0.

24 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

25 Jan van den Brand, Sebastian Forster, and Yasamin Nazari. Fast deterministic fully dynamic distance approximation. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 1011–1022. IEEE, 2022. doi:10.1109/FOCS54457.2022.00099.

26 Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 456–480. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00036.

27 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.