Beating Matrix Multiplication for $n^{1/3}$-Directed Shortcuts

Shimon Kogan
Weizmann Institute
shimon.kogan@weizmann.ac.il

Merav Parter *
Weizmann Institute
merav.parter@weizmann.ac.il

Abstract

For an n-vertex digraph $G = (V, E)$ and integer parameter D, a *D-shortcut* is a small set H of directed edges taken from the transitive closure of G, satisfying that the diameter of $G \cup H$ is at most D. A recent work [Kogan and Parter, SODA 2022] presented shortcutting algorithms with improved diameter vs. size tradeoffs. Most notably, obtaining linear size D-shortcuts for $D = \widetilde{O}(n^{1/3})$, breaking the $\sqrt{n}$-diameter barrier. These algorithms run in $O(n^\omega)$ time, as they are based on the computation of the transitive closure of the graph.

We present a new algorithmic approach for D-shortcuts, that matches the bounds of [Kogan and Parter, SODA 2022], while running in $o(n^\omega)$ time for every $D \geq n^{1/3}$. Our approach is based on a reduction to the min-cost max-flow problem, which can be solved in $\widetilde{O}(m + n^{3/2})$ time due to the recent breakthrough result of [Brand et al., STOC 2021].

We also demonstrate the applicability of our techniques to computing the minimal chain covers and dipath decompositions for directed acyclic graphs. For an n-vertex m-edge digraph $G = (V, E)$, our key results are:

- An $\widetilde{O}(n^{1/3} \cdot m + n^{3/2})$-time algorithm for computing D-shortcuts of linear size for $D = \widetilde{O}(n^{1/3})$, and an $\widetilde{O}(n^{1/4} \cdot m + n^{7/4})$-time algorithm for computing D-shortcuts of $\widetilde{O}(n^{3/4})$ edges for $D = \widetilde{O}(n^{1/2})$.
- For a DAG G, we provide $\widetilde{O}(m + n^{3/2})$-time algorithms for computing its minimum chain covers, maximum antichain, and decomposition into dipaths and independent sets. This improves considerably over the state-of-the-art bounds by [Caceres et al., SODA 2022] and [Grandoni et al., SODA 2021].

Our results also provide a new connection between shortcutting sets and the seemingly less related problems of minimum chain covers and the maximum antichains in DAGs.

*This project is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 949083).

Contents

1 Introduction

This paper is concerned with time-efficient algorithms for computing directed shortcut sets. For a given n-vertex digraph $G = (V, E)$ and an integer parameter D, a D-shortcut set H is a subset of edges from the transitive closure of G, denoted as $TC(G)$, satisfying that the diameter of $G \cup H$ is at most D. The diameter of the digraph is length of the longest u-v shortest path in G over any pair $(u, v) \in TC(G)$. The key objective in this setting is to optimize the diameter vs. size tradeoff. Since their introduction by Ullman and Yannakakis [UY91] and Thorup [Tho92], shortcutting sets have been studied extensively due to their wide-range of applications for parallel, distributed, dynamic and streaming algorithms [KS97, HKN15, FN18, LJS19, Fin20, GW20, BGW20]. Their applicability is also demonstrated by the recent breakthrough results, e.g., [LJS19, Fin20, GW20] that use shortcuts as their core component.

Diameter vs. Size Tradeoffs of Shortcuts. Thorup conjectured [Tho92] that any n-vertex digraph with m edges, has an D-shortcut of size $\widetilde{O}(m)$ for[1] $D = \widetilde{O}(1)$. This has been shown to hold for a restricted class of graphs, such as planar [Tho92]. Hesse [Hes03] refuted this conjecture for general graphs by presenting a construction of an n-vertex digraph with $m = \Theta(n^{19/17})$ edges that requires $\Omega(mn^{1/17})$ shortcut edges to reduce its diameter to below $n^{1/17}$. Up to very recently, the only known size vs. diameter tradeoff was given by a folklore randomized algorithm, attributed to Ullman and Yannakakis [UY91], that for every integer $D \geq 1$, provides a D-shortcut of size $O((n/D)^2 \log^2 n)$. Berman et al. [BRR10] improved the size bound to $O((n/D)^2)$. Setting $D = O(\sqrt{n})$ provides D-shortcuts of linear size in the number of vertices of the graph. This should be compared with the lower bound result by Huang and Pettie [HP18] that admits an n-vertex graph for which any linear shortcut (in the number of vertices of the graph) provides a diameter of $\Omega(n^{1/6})$. Lu, Vassilevska-Williams, Wein and Xu have recently improved the lower bound for D-shortcuts with $O(m)$ edges to $D = \Omega(n^{1/8})$.

Very recently, the authors [KP22] presented an improved tradeoff, breaking the $\sqrt{n}$ diameter barrier for linear-size shortcuts. For any $D \leq n^{1/3}$, they provide D-shortcuts of $\widetilde{O}(n^2/D^3)$ edges. For any $D > n^{1/3}$, they provide D-shortcuts with $\widetilde{O}((n/D)^{3/2})$ edges, hence of sublinear size. The algorithms of [KP22] are critically based on the precomputation the transitive closure of the graph, whose $O(n^\omega)$-time computation dominates the runtime of their shortcut constructions.

Time-Efficient Shortcut Computation. Due to the algorithmic importance of shortcuts, e.g., for reachability computation, much focus has been devoted to their time-efficient computation, in almost any classical computational setting, e.g., sequential, parallel and distributed, etc. In this algorithmic context, the primary objective is to compute shortcuts *faster* than computing the transitive closure itself. The reason is that shortcuts are usually used to provide faster algorithms for reachability related problems. The latter can be trivially solved by computing the transitive closure. Hence, for shortcuts to become algorithmically applicable, say in the sequential setting, their construction should run in $o(n^\omega)$ time. In a sequence of breakthrough results, Liu, Jambulapati and Sidford [LJS19] extended the framework of Fineman [Fin20] to compute, in nearly linear time, a shortcut set of $\widetilde{O}(n)$ edges that reduces the diameter of the graph to $D = O(n^{1/2+o(1)})$ (i.e., almost as obtained by the folklore algorithm). They also provide a parallel implementation of their algorithm leading to the first parallel reachability algorithm with near linear work

[1] The $\widetilde{O}(.)$ notation hides poly-log n factors.

and $n^{1/2+o(1)}$ depth. The algorithms of [KP22] are based on computing the transitive closure of the graph. Towards making these structures applicable in the algorithmic context, e.g., for reachability, we ask:

Question 1.1. *Is it possible to break the $\sqrt{n}$ diameter barrier of shortcuts in $o(n^\omega)$ time?*

We answer this question in the affirmative by providing a new algorithmic approach for shortcuts, that is based on a reduction to the *minimum-cost maximum-flow* problem. Luckily, the time complexity of the latter problem has only recently improved from $\widetilde{O}(\sqrt{n}m)$ (by Lee and Sidford [LS14]) to $\widetilde{O}(m+n^{3/2})$ by Brand et al. [vdBLL+21]. We observe that the key algorithmic challenge is in computing a collection of ℓ vertex-disjoint dipaths $\mathcal{P}$ (possibly in $TC(G)$) satisfying that the length of the longest path contained in the transitive closure induced on the set of *uncovered* vertices, namely, $U = V \setminus \bigcup_{P\in\mathcal{P}} V(P)$, is bounded by $\Theta(n/\ell)$. Our min-cost max-flow based approach for computing these dipaths finds broader applications for the seemingly less related problems of minimum chain covers and maximum antichains in directed acyclic graphs. Finally, we note our time complexity of shortcuts cannot be expressed *only* as a function of the time complexity of the min-cost max-flow problem, due to other computational steps that require $\widetilde{O}(m+n^{3/2})$ time.

Chains and Antichains: Covers and Decompositions. A *chain* is a dipath in the transitive closure $TC(G)$, and an antichain is subset of vertices $I \subseteq V$ that form an independent set in $TC(G)$. The closely related notions of chains and antichains give rise to four classical graph theoretical problems that have been studied thoroughly in the literature:

- **Minimum Chain Covers (MCC):** A minimum set of chains covering all vertices in G.
- **Maximum Antichain (MA):** An antichain of maximum size[2].
- **Minimum Antichain Covers (MAC):** A minimum set of antichains covering all $V(G)$.
- **Longest Chain (LC):** A dipath of largest length in $TC(G)$.

The cardinality of the MCC is also known as the *width* of the graph, denoted by $\omega(G)$. Dilworth [Dil50] showed that the size of the MA is the same as the size of the MCC, namely, $\omega(G)$. In a somewhat dual manner, Mirsky [Mir71] showed that the length of the LC equals to the size of the MAC. Interestingly, while there are linear time algorithms for computing the MAC and LC (e.g., by dynamic programming, [Mir71]), no such algorithms are known for the MCC and MA problems, at least for graphs with *non-constant* width. To date, the time complexity of the existing algorithms has an inherent dependency in $\omega(G)$. For example, for the MCC problem, the classical approach computes the chains one-by-one by solving $\omega(G)$ max-flow instances. Recent work by Cáceres et al. [CCM+21a, CCM+21b] has studied MCC algorithms whose complexity is parametrized on $\omega(G)$. The state-of-the-art is due to [CCM+21b] that runs in $\widetilde{O}(\omega(G)^2 n+m)$ time (for solving both MCC and MA). In light of the current gap in time complexity gap for solving MAC and LC on the one hand, and for MCC and MA, on the other hand, we ask:

Question 1.2. *Is it possible to* also *compute MCC and MA in (near) linear time?*

[2] I.e., maximum independent set in $TC(G)$.

Computation time	Citation
$O(n^3)$	[Jag90]
$O(n^2+\omega(G)^{3/2}n)$	[CC08]
$O(\max(m\sqrt{n},\omega(G)^{3/2}n))$	[CC14]
$O(\omega(G)m\log n)$	[MTK+19](*)
$O(\omega(G)^2 n\log n+m)$	[CCM+21b](*)
$\widetilde{O}(m+n^{3/2})$	**(this paper)**

Table 1: Prior work on MCC. $(*)$ indicates that the MCC result is implicit.

We answer this question, at least partially, in the affirmative by presenting linear time MCC and MA algorithms for moderately dense graphs. It is also known that MCC and MA are at least as hard as computing maximum matching in bipartite graphs. Interestingly, our algorithms use the shortcut algorithms in a black-box manner. An interesting take-home message is that the MCC and MA computations can become faster by reducing first the the diameter of the graph.

1.1 Our Results

We provide new algorithms for diameter shortcutting that break the $\sqrt{n}$ barrier without using any matrix multiplication, running in $o(n^\omega)$ time[3]. For example, for sparse graphs, our algorithms run in $o(n^2)$ for any diameter $D=\widetilde{\Omega}(1)$. Our first result, which in fact also serves as a building block later on, computes $\sqrt{n}$-shortcuts in near linear time:

Theorem 1.3. *[$\Theta(\sqrt{n})$-Shortcuts of Linear Size] There exists a randomized algorithm that for every n-vertex digraph G with m edges computes, w.h.p., in time $\widetilde{O}(m+n^{3/2})$ a shortcut set $H\subseteq TC(G)$ satisfying that $|E(H)|=\widetilde{O}(n)$ and $\texttt{Diam}(G\cup H)=O(\sqrt{n})$.*

For dense graphs, with $\Omega(n^{3/2})$ edges, this improves the diameter bound of Liu, Jambulapati and Sidford [LJS19] that yields D-shortcuts for diameter $D=n^{1/2+o(1)}$. We next use this improved algorithm for computing D-shortcuts for any $D=O(\sqrt{n})$, to provide the following:

Theorem 1.4. *[$o(\sqrt{n})$-Shortcuts] There exists a randomized algorithm that for every n-vertex m-edge digraph G and $D=O(\sqrt{n})$, computes, w.h.p., in time $\widetilde{O}(m\cdot n/D^2+n^{3/2})$ a D-shortcut set $H\subseteq TC(G)$ with $|E(H)|=\widetilde{O}(n^2/D^3+n)$ edges.*

For example, for $D=n^{1/3}$, we get D-shortcuts of near-linear size in $\widetilde{O}(m\cdot n^{1/3}+n^{3/2})$ time, improving on the time complexity of $O(n^\omega)$ by [KP22].

We also consider faster algorithms for computing shortcuts with sublinear number of edges. This turns out to be quite technically involved, as many of the tools used in Theorem 1.4 are based on adding a *linear* number of edges. We show:

Theorem 1.5. *[$\omega(n^{1/3})$-Shortcuts of Sublinear Size] There exists a randomized algorithm that for every n-vertex m-edge digraph G and $D=\omega(n^{1/3})$, computes, w.h.p., in time $\widetilde{O}((m+n^{3/2})\sqrt{n/D})$ a D-shortcut set $H\subseteq TC(G)$ with $|E(H)|=\widetilde{O}((n/D)^{3/2})$ edges.*

For example, for $D=n^{1/2}$, this provides D-shortcuts with $\widetilde{O}(n^{3/4})$ edges in $\widetilde{O}(m\cdot n^{1/4}+n^{7/4})$ time, improving on the time complexity of $O(n^\omega)$ by [KP22].

[3]The state-of-the-art value for the matrix multiplication constant is $\omega=2.372$ due to Alman and Vassilevska Williams [AW21]

Application to Decompositions and Covers of Chains and Antichains. Recall that a chain (resp., antichain) is a dipath (resp., independent set) in $TC(G)$. A minimum chain cover (MCC) for a DAG $G = (V, E)$ is a collection of vertex-disjoint chains that cover $V(G)$. The maximum antichain is the maximum independent set in $TC(G)$. We provide the first nearly linear algorithms for moderately dense graphs, that in contrast to prior algorithms, do not depend on the width of the graph.

Theorem 1.6. *For every n-vertex m-edge DAG $G = (V, E)$, one can compute the minimum chain cover, as well as, the maximum antichain in time $\widetilde{O}(m + n^{3/2})$, w.h.p.*

This improves over that very recent state-of-the-art running time of $O((\omega(G))^2 n \log n + m)$ by [CCM+21b] for graphs of large width $\omega(G) = \widetilde{\Omega}(n^{1/4})$. More conceptually this provides a near linear-time algorithm for dense graphs, narrowing the gap to the related problems of MAC and ML for which linear time algorithms are folklore. Finally, we consider the decomposition of G into a collection of vertex-disjoint dipaths $\mathcal{P}$ and vertex-disjoint independent sets[4].

Definition 1.1. [Slight restatement of Def. 3.1 of [GIL+21]] An (a, b)-decomposition of a directed acyclic graph (DAG) G consists of a collection $\mathcal{P} = \{P_1, \ldots, P_a\}$ of a vertex-disjoint directed paths in G, and a collection $\mathcal{Q} = \{Q_1, \ldots, Q_b\}$ of b vertex-disjoint independent sets of G such that $\bigcup_{i=1}^{a} V(P_i) \cup \bigcup_{j=1}^{b} Q_j = V(G)$.

Grandoni et al. [GIL+21] presented an $O(n^2)$-time algorithm for computing an $(\ell, 2n/\ell)$ decomposition in G. In fact, the algorithm of [KP22] employed this decomposition on the transitive closure of G. We observe that by reducing to the min-cost max-flow problem, one can break the $O(n^2)$-time complexity. We show:[5]

Theorem 1.7. *Let $G = (V, E)$ be an n-vertex DAG. For every $\ell \in [1, n]$, there is an $\widetilde{O}(|E| + n^{3/2})$-time randomized algorithm for computing an $(\ell, 2 \cdot n/\ell)$-decomposition of G.*

1.2 Technical Overview

1.2.1 A New Algorithmic Approach for Shortcuts

For the sake of presentation, we focus on the computation of a $D = O(n^{1/3})$-shortcut H of near-linear size. We start by providing a succinct description of the our algorithm in [KP22]. Throughout, we assume, w.l.o.g., that the input graph G is a DAG, and denote its transitive closure by $TC(G)$. We note that in context of computing shortcuts of *sublinear* size, we provide an alternative reduction to DAG[6]. For a dipath P, let $H(P)$ be a 2-shortcut set for P satisfying that $\texttt{Diam}(P \cup H(P)) \leq 2$. It is well-known that one can compute $H(P)$ in nearly-linear time and that $|H(P)| = O(|P| \log |P|)$. For a path collection $\mathcal{P}$, let $V(\mathcal{P}) = \bigcup_{P \in \mathcal{P}} V(P)$. Finally, for a set of elements X and probability $p \in [0, 1]$, let $X[p]$ be the subset obtained by sampling each element of X into $X[p]$ independently w.p. p.

[4]Grandoni et al. [GIL+21] called this *chain and antichain decomposition*. Since this decomposition is in G and the notions of chain and antichains are usually defined in $TC(G)$, we use the terminology of dipaths and independent sets.

[5]We provide this as an independent observation. As this decomposition is w.r.t G (rather than $TC(G)$), it is not useful for our shortcut algorithms.

[6]An alternative reduction is needed since the folklore approach of contracting each strongly connected component in G introduces linear number of edges to the shortcut.

A Quick Recap of the $O(n^\omega)$-Time Algorithm by [KP22]. The algorithm starts by computing $TC(G)$ in $O(n^\omega)$-time. It then applies the decomposition algorithm Grandoni et al. [GIL+21] to partition $V(TC(G))$ into $n^{2/3}$ vertex-disjoint dipaths $\mathcal{P}$ (denoted as chains), and $2 \cdot n^{1/3}$ vertex-disjoint antichains $\mathcal{Q} = \{Q_1, \ldots, Q_k\}$. I.e., $V = V(\mathcal{P}) \cup V(\mathcal{Q})$. Letting $U = V(\mathcal{Q})$, the key property that we use in the diameter argument is that the length of a longest path of the graph[7] $TC(G)[U]$ is bounded by $k = n^{1/3}$, which indeed follows by the definition of $\mathcal{Q}$.

For a vertex v and a dipath P, let $e(v, P)$ be the edge connecting v to the *first* vertex in $V(P)$ that is reachable from v (if such exists). Let $\mathcal{P}' = \mathcal{P}[p]$ and $V' = V[p]$ for $p = n^{-1/3}$ (i.e., subsamples of dipaths and vertices). Then the output shortcut set is $H = H_1 \cup H_2$, where $H_1 = \bigcup_{P \in \mathcal{P}} P \cup H(P)$ and $H_2 = \{e(v, P) \mid v \in V', P \in \mathcal{P}'\}$. Summarizing, the key algorithmic steps can be summarized by:

- **Step 1.** Computing a collection $\mathcal{P}$ of (at most) $n^{2/3}$ *vertex-disjoint* dipaths such that the length of the longest dipath of $TC(G)[U]$ is of length $O(n^{1/3})$, where $U = V \setminus V(\mathcal{P})$.
- **Step 2.** Computing the edges $e(v, P)$ for every $v, P \in \mathcal{P}' \times V'$.

The implementation of [KP22] for both steps uses $TC(G)$ in a strong manner. Before explaining our approach, we sketch the size and diameter bounds of the above construction.

Size and Diameter Bound. The size bound is immediate: the $\mathcal{P}$ are vertex-disjoint, thus $|H_1| = \widetilde{O}(n)$. W.h.p., $|V'| = \widetilde{O}(n^{2/3})$ and $|\mathcal{P}'| = \widetilde{O}(n^{1/3})$, therefore, $|H_2| = \widetilde{O}(n)$, as well.

Consider a u-v shortest path P in $G \cup H_1$ and denote its $7n^{1/3}$-length prefix (resp., suffix) by P', P'', respectively. By Chernoff, w.h.p., there exists some sampled vertex $u^* \in V(P') \cap V'$. The challenge is in showing that P'' intersects with $\Omega(n^{1/3})$ distinct paths in $\mathcal{P}$, and thus, w.h.p., intersects with at least one sampled path in $\mathcal{P}'$. This is shown by observing that (i) P'' contains at most 3 vertices from each dipath $Q \in \mathcal{P}$ (due to the addition of the shortcut $H(Q)$), and that (ii) P'' contains at most $n^{1/3}$ vertices in U. Altogether, we have some $Q^* \in \mathcal{P}'$ satisfying that $V(Q^*) \cap V(P'') \neq \emptyset$. The edge $e(u^*, Q^*) \in H_2$ then provides us an $O(n^{1/3})$-hop length dipath from u^* to $w \in V(Q^*) \cap V(P'')$.

New Approach: Shortcuts via Min-Cost Max-Flows. We focus on Step (1), as Step (2) can be implemented in $O(m \cdot n^{1/3})$ time using e.g., dynamic programming. Our goal is to show that Step (1) can be implemented in $\widetilde{O}(m + n^{3/2})$ time, which establishes Theorem 1.4 for $D = n^{1/3}$. The challenge boils down into computing a *nice* collection of dipaths which also has further independent applications, as illustrated in this paper. Denote by $\mathrm{LP}(G)$ to be the length of a longest simple dipath in G.

> **Key Task:** Compute in $o(n^\omega)$-time a collection of (at most) $n^{2/3}$ vertex-disjoint dipaths $\mathcal{P}$ in $TC(G)$, such that $\mathrm{LP}(G') = O(n^{1/3})$ where $G' = TC(G)[U]$ and $U = V \setminus V(\mathcal{P})$.

Note that the above requires to bound the length of the longest path in the graph[8] $TC(G)[U]$. This task can also be viewed as a relaxed chain and antichain decomposition (see Def. 1.1) for $\ell = n^{2/3}$, where it is required to output only chain part, while guaranteeing the remaining

[7] I.e., the transitive closure $TC(G)$ induced on the uncovered vertices U, namely, vertices that do not appear on $\mathcal{P}$.
[8] The transitive closure of G induced on the vertices of U.

uncovered vertices, U, can be decomposed into $n^{1/3}$ anti-chains[9]. Clearly the challenge is that it is required to compute such (partial) decomposition in $TC(G)$ *without ever* computing it explicitly! While our goal is to solve it in $\widetilde{O}(m+n^{3/2})$ time, the decomposition algorithm of [GIL+21] runs in $O(n^2)$ even when *given* $TC(G)$. Our challenge is therefore two-folds.

We introduce ℓ-*covers* which for a given input parameter $\ell \in [1,n]$ provide a *multiset* $\mathcal{P} = \{P_1,\ldots,P_\ell\}$ of ℓ dipaths in G. Letting, $\text{TotLen}(\mathcal{P}) = \sum_{P_i\in\mathcal{P}} |P_i|$ and $U = V \setminus V(\mathcal{P})$ (i.e., the total dipath lengths), then the ℓ-cover satisfies:

1. $\text{TotLen}(\mathcal{P}) \leq \min\{\ell, \texttt{Diam}(G)\} \cdot n$,
2. $\text{LP}(G') \leq 2n/\ell$ where $G' = TC(G)[U]$.

Note that an ℓ-cover for $\ell = n^{2/3}$, almost fits the key task, up to one major caveat: the total path length, $\text{TotLen}(\mathcal{P})$, might be super-linear, hence leading to shortcuts of super-linear size[10]. We put this technicality aside for a moment, explain first how to compute ℓ-covers *without* computing $TC(G)$, and then show how to use them to solve the key task.

Computing ℓ-Covers in Time $\widetilde{O}(m+n^{3/2}+\min\{\ell,\texttt{Diam}(G)\}\cdot n)$. The algorithm is based on a reduction to the min-cost max-flow problem. We define a flow-instance $\widetilde{G} = (\widetilde{V},\widetilde{E},u,c)$ corresponding to G, where $|\widetilde{V}| = O(n), |\widetilde{E}| = O(m)$, $u \in \mathbb{Z}^{\widetilde{E}}_{\geq 0}$ and $c \in \mathbb{Z}^{\widetilde{E}}$ are the capacity and cost functions, respectively. We then apply the recent algorithm of van den Brand et al. [vdBLL+21] to compute the min-cost max-flow s-t flow in $\widetilde{G}$ for a given pair $s,t \in \widetilde{V}$. In their breakthrough result, [vdBLL+21] provided an $\widetilde{O}(m+n^{3/2})$-time randomized algorithm for this problem, provided that the edge capacities and costs are *integral*.

The desired ℓ-cover collection is then obtained by computing the flow-decomposition of the output integral flow. Since G is a DAG, this decomposition can be done in linear time in the total length of the output path collection. Finally, we translate the collection of dipaths in $\widetilde{G}$ into a collection of dipaths in G, which provides the desired ℓ-cover $\mathcal{P}$. Importantly, our flow-instance deviates from the classical flow-reductions[11] by incorporating a special gadget per vertex $v_i \in V$. We carefully set the capacity and the cost values in a way that on the one hand, allows a given vertex to participate in multiple paths, while at the same time bounding the total length of $\mathcal{P}$, and most importantly bounds the longest path length of the transitive closure induced on the uncovered vertices U, $TC(G)[U]$. Altogether, the computation of the ℓ-cover $\mathcal{P}$ takes $\widetilde{O}(m+n^{3/2}+\text{TotLen}(\mathcal{P}))$ time which by Property (1) is bounded by $\widetilde{O}(m+n^{3/2}+n\min\{\ell,\texttt{Diam}(G)\})$.

ℓ-Covers in $\widetilde{O}(m+n^{3/2})$ Time. To beat this dependency in $\text{TotLen}(\mathcal{P})$, we take the following strategy. First, we reduce the diameter of G to $d = O(\sqrt{n})$ by computing a d-shortcut set H_0 of linear size. We show that this can be done using ℓ-covers for $\ell = \sqrt{n}$, hence taking $O(m+n^{3/2})$ time. We then apply the ℓ-cover algorithm on the graph $G \cup H_0$. Since the diameter of $G \cup H$ is $O(\sqrt{n})$, we have that $\text{TotLen}(\mathcal{P}) = O(n^{3/2})$, and the entire computation takes $\widetilde{O}(m+n^{3/2})$ time.

Implementing Step (1) in $\widetilde{O}(m+n^{3/2})$ Time. So-far, we have computed a collection of $O(n^{2/3})$ dipaths $\mathcal{P}$ (in $G \cup H_0$) of total length $O(n^{3/2})$, and such that the uncovered diameter is at most $O(n^{1/3})$. It remains to turn $\mathcal{P}$ into vertex-disjoint dipaths in $TC(G)$. This can be easily done by iterating over the dipaths of $\mathcal{P}$, one by one, shortcutting the given dipath by cutting out

[9] By [Mir71], a bounded longest path of $TC(G)[U]$ indeed guarantees the antichain decomposition of U.

[10] As we include in the output shortcut, the union of path shortcuts $H(P)$ for every $P \in \mathcal{P}$.

[11] E.g, where each vertex $v_i \in V$ has two copies v_i^{in} and v_i^{out} connected by a directed edge of capacity 1.

the vertices that appear in the currently collection of vertex-disjoint dipaths. This works in $O(\text{TotLen}(\mathcal{P})) = O(n^{3/2})$ time.

Shortcuts of Sublinear-Size. In [KP22], it was shown that for any $D = \omega(n^{1/3})$, one can compute D-shortcuts of size $\widetilde{O}((n/D)^{3/2})$. It will be instructive to consider the case where $D = O(\sqrt{n})$ for which the construction provides shortcuts with $\widetilde{O}(n^{3/4})$ edges. The algorithm of [KP22] for this setting is based on sampling a collection of landmarks $V' = V[p]$ for $p = n^{1/4}$. It then defines the graph $G' = (V', E')$ where $E' = \{(u, v) \mid \text{dist}_G(u, v) \leq n^{1/4}\}$, and applies the above mentioned algorithm for computing $D' = |V'|^{1/3}$-shortcut H of size $O(|V'|)$. Since $|V'| = O(n^{3/4})$, w.h.p., this provides a shortcut of $O(n^{3/4})$ edges.

In our context, computing the graph G' takes $O(|V'|m)$ time which is too costly for our purposes. Another challenge in provide a *sublinear* reduction into DAGs. So-far, we assumed w.l.o.g. that G is a DAG, by using the following folklore reduction: shortcut each strongly connected component C to diameter 2, by connecting the all vertices of C to center vertex $v \in C$. This allows one to restrict attention to the DAG obtained by contracting each component. This reduction, however, adds $\Theta(n)$ shortcut edges, which is above our budget. Our algorithm provides an alternative reduction. Given G, we compute a DAG G' of similar size and show that any D-shortcut set H' for G' can be translated into a D-shortcut set H for G, where $|H| = O(|H'|)$. This allows us to restrict attention to DAGs. We believe this alternative reduction should be of general interest.

Our shortcut algorithm calls for a collection of additional refined tools. We introduce partial ℓ-covers that provide the desired properties (1,2), up to some slack, with respect to a given *subset* of vertices. The shortcut algorithm is then based on iteratively computing collections of partial ℓ-covers, to mimic, in some way, the effect of the single application of a shortcut computing on the virtual graph G'.

1.2.2 Minimum Chain Covers, Maximum Antichain, and More

Recall that a chain is a dipath in $TC(G)$ and an antichain is an independent set in $TC(G)$. The cardinality of the minimum chain cover (MCC) of a DAG G is denoted as the width of G, $\omega(G)$. An minimum path cover (MPC) is a minimum set of dipaths (not necessarily disjoint) in G that cover V. Prior algorithms for computing the MCC are based on iteratively solving a maximum s-t flow instance in the residual graph, each iteration provides one additional dipath to the output collection. This dipath-by-dipath approach requires solving $\omega(G)$ flow computations. Note that computation of the value of the width, $\omega(G)$, can be easily done using only $O(\log n)$ max-flow computations (by binary search). Hence, the main challenge is in the computation of chain cover itself.

Our approach is based on having a single application of the min-cost max-flow algorithm. The starting observation is that thanks to our shortcut algorithms, we can assume, w.l.o.g., that the diameter of G is $O(\sqrt{n})$. This is because we can apply the MCC algorithm on the graph $G \cup H_0$ where H_0 is the $O(\sqrt{n})$-shortcut set for G, obtained by Thm. 1.3. Since we aimed at computing *chains*, rather than dipaths in G, we can safely work on $G \cup H_0$.

Using our flow reduction, we are able to compute a collection $\mathcal{P}$ of $\omega(G)$ dipaths (these are dipaths in $G \cup H_0$), in time $\widetilde{O}(m + n^{3/2} + \text{TotLen}(\mathcal{P}))$. By the way that we set the cost of the edges in our instance, $\mathcal{P}$ covers all vertices in G. I.e., it is an MPC in $G \cup H_0$. This by itself is not enough

as the dipaths of $\mathcal{P}$ are might not be vertex-disjoint, possibly with large total length, TotLen($\mathcal{P}$). To overcome this, we first provide an algorithm for computing a dipath cover of *minimum* total length. We introduce the notion of *minimum-length cover* (MLC): a collection of $\omega(G)$ dipaths, that covers all vertices while minimizing the *total length* of the dipaths.

Then, we provide a combinatorial argument that shows that any n-vertex DAG admits an MPC $\mathcal{P}$ of length TotLen($\mathcal{P}$) $\leq n \cdot \texttt{Diam}(G)$. Since $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$, we end up with having a collection of dipaths of total length $O(n^{3/2})$. These dipaths can be transformed in vertex-disjoint chains in time $O(n^{3/2})$, in a brute-force manner (Lemma 1.12). This completes the high-level approach.

1.3 Preliminaries

Graph Notations. For an n-vertex digraph G, let $TC(G)$ denote its transitive closure. We also denote $TC(G)$. For a vertex pair $u, v \in V(G)$ where $(u, v) \in TC(G)$, let $\text{dist}_G(u, v)$ be the length (measured by the number of edges) of the shortest dipath from u to v. For $(u, v) \notin TC(G)$, $\text{dist}_G(u, v) = \infty$. For a subset $V' \subseteq V$, let $G[V']$ be the induced graph on V'. The graph *diameter* is denoted by $\texttt{Diam}(G) = \max_{(u,v)\in TC(G)} \text{dist}_G(u, v)$. We say that $u \rightsquigarrow_G v$ if there is a directed path from u to v in G, i.e., $(u, v) \in TC(G)$. A *shortcut* edge is an edge in $TC(G)$. For a vertex $v \in G$, let $N_{in}(v, H) = \{u \mid (u, v) \in G\}$ denote the incoming neighbors of v in G. The set of outgoing neighbors $N_{out}(v, G)$ is defined in an analogous manner. For a collection of paths $\mathcal{P}$, the vertices of $\mathcal{P}$ is denoted by $V(\mathcal{P}) = \bigcup_{P\in\mathcal{P}} V(P)$. For a dipath $P = [u_0, \ldots, u_k] \subseteq G$, for $j \leq q$, let $P[u_j, u_q]$ denote the path segment $P[u_j, u_{j+1}, \ldots, u_q]$. In the case where $u_j = u_0$ (resp., $u_q = u_k$), we simply write $P[\cdot, u_q]$ (reps., $P[u_j, \cdot]$). For an a-b dipath P and an b-c dipath P' the concatenation of the paths is denoted by $P \circ P'$. Let $|P|$ denote the number of edges on P (unless mentioned otherwise). For a set of elements X and $p \in [0, 1]$, let $X[p]$ be the set obtained by taking each element of X into $X[p]$ independently with probability p.

Definition 1.2. For a digraph G, $H \subseteq TC(G)$ is *D-shortcut* if $\texttt{Diam}(G \cup H) \leq D$.

The following lemma computes linear 2-shortcuts for directed paths:

Lemma 1.8. *[Lemma 1.1, [Ras10]] Given a directed path P, one can compute in $O(|P| \log |P|)$ time, a 2-shortcut $H(P)$ for P where $|H| = O(|P| \log |P|)$ edges.*

Given a digraph G, by G^+ we denote the result of contracting all strong components in G. Note that G^+ is acyclic directed graph (DAG) and can be computed in linear time.

Proposition 1.9. *[Lemma 3.2, [Ras10]] Let H^+ be a shortcut set for G^+. One can compute in $O(|E(G)| + |H|)$-time, a shortcut set H for G such that $|H| = O(|H^+| + n)$ and $\texttt{Diam}(G \cup H) = O(\texttt{Diam}(G^+ \cup H^+))$.*

Algorithmic Tools for Flow Computation. In the *minimum-cost maximum-flow* problem, given is a connected directed graph $G = (V, E, u, c)$ with edges capacities $u \in \mathbb{R}^E_{\geq 0}$ and costs $c \in \mathbb{R}^E$ (which can be negative). The vector $x \in \mathbb{R}^E$ is an *s-t flow* for $s, t \in V$ if $x(e) \in [0, u(e)]$ for all e in E, and for each vertex $v \notin \{s, t\}$ the amount of flow entering v equals the amount of flow leaving v, i.e., $\sum_{e=(a,v)} x(e) = \sum_{e=(v,b)} x(e)$. The *cost* of a flow x is defined by $c(x) = \sum_e c(e)x(e)$. The *value* of an s-t flow is the amount of flow leaving s, i.e., $val(x) = \sum_{e=(s,v)} x_e$ (or equivalently, entering t, i.e., $\sum_{e=(v,t)} x_e$). The objective is to compute a maximum s-t flow of minimum cost denoted by $\sum_{e\in E} c_e x_e$. The following theorem was proven in [vdBLL+21].

Theorem 1.10. *[Theorem 1.4 of [vdBLL+21]] There is an algorithm* MinCostFlow(G, s, t) *that, given a n-vertex m-edge digraph $G = (V, E, u, c)$, integral edge capacities $u \in \mathbb{Z}^E_{\geq 0}$ and costs $c \in \mathbb{Z}^E$, w.h.p., computes an integral minimum cost maximum flow in time*

$$\widetilde{O}\left(m \log(||u||_\infty ||c||_\infty) + n^{3/2} \log^2(||u||_\infty ||c||_\infty)\right) .$$

We observe that even more recently, there have been additional breakthrough results [AMV21, vdBGJ+21] that provide improved running time for sparse graphs, this however, does not effect our final time complexity.

Definition 1.3. For a digraph $G = (V, E)$ and a given valid s-t flow vector $x \in \mathbb{N}^E_{\geq 0}$, a *flow decomposition* is a multiset of s-t dipaths in G given by $\mathcal{Q} = \{P_1, \ldots, P_k\}$, such that for every $e \in E$, it holds that $x(e) = |\{P_i \in \mathcal{Q} \mid e \in P_i\}|$. I.e., $x(e)$ equals the number of paths in the multiset containing e.

Lemma 1.11. *Let $G = (V, E)$ be an n-vertex m-edge DAG and let $x \in \mathbb{N}^E_{\geq 0}$ be an integral s-t flow of value k for $s, t \in V$, then one can compute a flow decomposition of x, denoted by $\mathcal{Q}$ in $O(m + n + \sum_{e \in E} x(e))$ time.*

Dipaths to Chains. Throughout, we make use of the following procedure to translate a collection of dipaths in G into a collection of vertex-disjoint chains.

Lemma 1.12. *Let $\mathcal{P}$ be a collection of dipaths in G, there is an algorithm* DisjointChains *that in time $O(\sum_{P \in \mathcal{P}} |P|)$ computes a collection of* vertex-disjoint *chains $\mathcal{C}$ satisfying that: (i) $|\mathcal{C}| \leq |\mathcal{P}|$, (ii) $V(\mathcal{C}) = V(\mathcal{P})$ and (iii) every $C \in \mathcal{C}$ is a dipath in $TC(G)$.*

Roadmap. In Sec. 2, we introduce ℓ-covers, that serve the key algorithmic part for computing our shortcuts. In Sec. 3.1, we compute $\sqrt{n}$-shortcuts of linear size (proof of Thm. 1.3). In Sec. 3.2, we provide an algorithm that computes $n^{1/3}$-shortcut of linear size in $\widetilde{O}(mn^{1/3} + n^{3/2})$ time, and more generally prove Thm. 1.4. Then Sec. 3.3 considers the constructions of shortcuts with sublinear number of edges establishing Thm. 1.5.

2 New Notions of Dipath Decompositions

2.1 ℓ-Covers

We introduce a new notion of dipath decomposition which we denote by *ℓ-covers*. For a given parameter ℓ, the ℓ-cover is given by a multiset of ℓ dipaths in G, $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, whose vertices $V(\mathcal{P})$ dominate long dipaths in G, in a way that becomes formal in Def. 2.1. As we will see, this notion is strong enough to substitute the path collection obtained by computing the chain-antichain decomposition in $TC(G)$. Missing proofs of this section are deferred to the full version. Recall that for a multiset of dipaths $\mathcal{P}$, $\text{TotLen}(\mathcal{P}) = \sum_{P_i \in \mathcal{P}} |P_i|$.

Definition 2.1 (ℓ-Covers)**.** For a given n-vertex digraph $G = (V, E)$ and an integer parameter $\ell \in [1, n]$, an ℓ-cover for G is a multiset of ℓ paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ (possibly consisting of single vertices or empty), satisfying the following:

1. $\text{TotLen}(\mathcal{P}) \leq \min(\ell \cdot n, \texttt{Diam}(G) \cdot n)$,

2. For any dipath $P \subseteq G$, it holds that $|V(P) \cap (V \setminus V(\mathcal{P}))| \leq 2n/\ell$.

Property (2) implies that $\mathrm{LP}(TC(G)[U]) \leq 2n/\ell$, that is the following holds:

Corollary 2.1. *Each path in $TC(G)[V \setminus V(\mathcal{P}]$ contains at most $2n/\ell$ vertices.*

Theorem 2.2. *Let G be an n-vertex DAG with m edges, then for every $\ell \geq 1$, there is a randomized algorithm for computing an ℓ-cover of G, w.h.p, in time $\widetilde{O}(m + n^{3/2} + n \cdot \min\{\mathtt{Diam}(G), \ell\})$.*

For our purposes of computing shortcuts, we employ the algorithm of Thm. 2.2 only in settings where $\min\{\mathtt{Diam}(G), \ell\} = O(\sqrt{n})$, therefore spending only $\widetilde{O}(m + n^{3/2})$ time.

Path Decomposition via Min-Cost Max-Flow. The decomposition algorithm is based on a reduction to the *minimum cost maximum flow* problem (see Sec. 1.3). For the given DAG G, the algorithm computes a corresponding min-cost flow instance $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$. At the high level, for each $v_i \in V(G)$, the graph $\widetilde{G}$ contains three inter-connected copies $v_i^{in}, v_i^{out}, v_i'$. The out-copy of the incoming G-neighbors of v_i are connected to the in-copy of v_i. In addition, $\widetilde{V}$ includes also three additional vertices, s, s' and t, where s and t are the source and target vertices given to the min-cost flow algorithms. The precise definition of $\widetilde{G}$ are specified in the pseudocode below, see also Fig. 1.

Let $x \in \mathbb{N}_{\geq 0}^{|\widetilde{E}|}$ be the output flow solution for the instance $\widetilde{G}$ obtained by applying Alg. MinCostFlow from Theorem 1.10. The algorithm then decomposes the flow x into a multiset of s-t paths $\widetilde{\mathcal{P}} = \{P_1', \ldots, P_\ell'\}$. Since without loss of generality the flow values are integrals and since $\widetilde{G}$ is a DAG, the flow decomposition of x can be computed efficiently by applying Lemma 1.11. Finally, the output ℓ-cover $\mathcal{P}$ is obtained by mapping each $\widetilde{G}$-dipath $P_j' \in \widetilde{\mathcal{P}}$ into a corresponding G-dipath P_j. This mapping is done by applying Procedure Translate on every $P_j' \in \widetilde{\mathcal{P}}$. The output G-path $P_j = \mathsf{Translate}(P_j')$ is defined as follows. The vertices s, s' and t are omitted, and each appearance of a vertex $u_j \in v_i^{in}, v_i^{out}, v_i'$ is replaced by its G-vertex v_i, we omit multiple successive occurrences of a vertex in this translation. For example, for $P' = [s, s', v_i^{in}, v_i^{out}, v_j^{in}, t]$, $\mathsf{Translate}(P') = [v_i, v_j]$. For a path $P'' = [s, s', t]$, $\mathsf{Translate}(P'') = \emptyset$.

The ℓ-cover is then given by the multiset $\mathcal{P} = \{\mathsf{Translate}(P_j') \mid j \in \{1, \ldots, \ell\}\}$. We need the following definition. For $P_j \in \mathcal{P}$, denote $\mathsf{Translate}^{-1}(P_j) = P_j'$ where $P_j' \in \widetilde{\mathcal{P}}$. Note that while each path $P_j' \in \widetilde{\mathcal{P}}$ is mapped to a *unique* G-path (given by $\mathsf{Translate}(P_j')$), it might be the case that two distinct paths $P_i', P_j' \in \widetilde{\mathcal{P}}$ map to the same G-path. For example, $P_i' = [s, s', v_i^{in}, v_i', v_i^{out}, t]$ and $P_j' = [s, s', v_i^{in}, v_i^{out}, t]$ both translate to the single vertex $\{v_i\}$. Therefore, we consider the Translate function as a bijection from the *multiset* $\widetilde{\mathcal{P}}$ to the *multiset* $\mathcal{P}$. I.e., each path $P_i \in \mathcal{P}$ is uniquely mapped to a path $P_i' = \mathsf{Translate}^{-1}(P_i)$ in the flow decomposition $\widetilde{\mathcal{P}}$ and vice-versa.

Algorithm PathCover**:**

Input: An n-vertex DAG G and a parameter $\ell \in \mathbb{N}_{\geq 1}$.
Output: An ℓ-cover $\mathcal{P}$ of G.

1. **Transformation to Min-Cost Max-Flow Instance.** The instance $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$ with the designated source s and target t vertices are defined as follows.

   - For each vertex $v_i \in V(G)$, include three copies $v_i^{in}, v_i^{out}, v_i'$ in $\widetilde{V}$. These copies are connected by the edges $E_{0,i} = \{(v_i^{in}, v_i^{out}), (v_i^{in}, v_i'), (v_i', v_i^{out})\}$.
     In addition, add three designated vertices s, s' and t. So overall,

     $$\widetilde{V} = \{v_i^{in}, v_i^{out}, v_i' \mid v_i \in V(G)\} \cup \{s, s', t\} .$$

   - Connect the out-copy v_j^{out} of every incoming neighbor $v_j \in N_{in}(v_i, G)$ to v_i^{in}. In the same manner, connect the in-copy v_k^{in} of every $v_k \in N_{out}(v_i, G)$ to v_i^{out}. Formally,

     $$E_1 = \{(v_j^{out}, v_i^{in}) \mid v_j \in N_{in}(v_i, G)\} \text{ and } E_2 = \{(v_j^{in}, v_i^{out}) \mid v_j \in N_{out}(v_i, G)\} .$$

   - Let $E_3 = \{(s', v_i^{in}) \mid v_i \in V(G)\} \cup \{(v_i^{out}, t) \mid v_i \in V(G)\} \cup \{(s, s'), (s', t)\}$. Then

     $$\widetilde{E} = \bigcup_i E_{0,i} \cup E_1 \cup E_2 \cup E_3.$$

   - The edge capacities $u \in \mathbb{Z}_{\geq 0}^{\widetilde{E}}$ and costs $c \in \mathbb{Z}^{\widetilde{E}}$ are defined by:
     (a) $u((s, s')) = \ell$ and $u(v_i^{in}, v_i^{out}) = 1$ for every $v_i \in V$. All remaining edges e in $\widetilde{E}$ have capacity of $u(e) = \ell$.
     (b) $c(v_i^{in}, v_i^{out}) = -n^3$ and $c(v_i^{in}, v_i') = 1$ for every $v_i \in V$. All remaining edges e in $\widetilde{E}$ have cost of $c(e) = 0$.

2. Apply Algorithm $\mathsf{MinCostFlow}(\widetilde{G}, s, t)$ (Theorem 1.10), and let $x \in \mathbb{R}^{|\widetilde{E}|}$ be the output flow.

3. **Flow Decomposition.** Decompose $x \in \mathbb{R}^{|\widetilde{E}|}$ into s-t paths $\widetilde{\mathcal{P}}$ in $\widetilde{G}$ by Lem. 1.11.

4. The ℓ-cover is given by the multiset $\mathcal{P} = \{\mathsf{Translate}(P') \mid P' \in \widetilde{\mathcal{P}}\}$ (where $|\mathcal{P}| = \ell$ as each path corresponds to one unit flow in $\widetilde{G}$).

Analysis. For a path $P' \subseteq \widetilde{G}$, denote the cost of P' by $c(P') = \sum_{e \in P'} c(e)$. We start with observing that any path P' in the flow decomposition $\widetilde{\mathcal{P}}$ must have a non-positive cost. Intuitively, this holds as one can replace P', with the zero-cost path $[s, s', t]$ that has a sufficiently large capacity. Hence we have the following observation.

Observation 2.1. Every path $P' \in \widetilde{\mathcal{P}}$ has a non-positive cost, $c(P') \leq 0$.

Recall that $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ where P_i might be equal to P_j. A vertex v_i in a path $P_j \in \mathcal{P}$ is denoted as a *principal vertex* of P_j if its corresponding flow path $\mathsf{Translate}^{-1}(P_j) \in \widetilde{\mathcal{P}}$ (i.e., the j^{th}

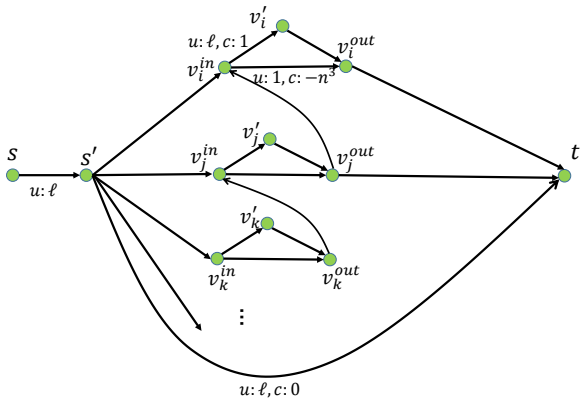

Figure 1: An illustration for the min-cost max-flow reduction. The default values for the edge capacities and costs are ℓ and 0, respectively.

path in $\widetilde{\mathcal{P}}$) contains the edge (v_i^{in}, v_i^{out}). We denote the number of principal vertices in $P_j \in \mathcal{P}$ by $n(P_j)$. Formally,

$$n(P_j) = |\{v_i \in V \mid (v_i^{in}, v_i^{out}) \in \mathsf{Translate}^{-1}(P_j)\} \,. \tag{1}$$

Since in our flow instance $\widetilde{G}$, the capacity of any edge $e = (v_i^{in}, v_i^{out})$ is 1, i.e., $u(e) = 1$, we have that a vertex $v_i \in V$ can be a principal vertex of at most *one* path in $\mathcal{P}$, hence:

Observation 2.2. $\sum_{P_j \in \mathcal{P}} n(P_j) \leq n$.

Lemma 2.3. $|\mathcal{P}| = \ell$ *and* $\mathrm{TotLen}(\mathcal{P}) \leq \min(\ell \cdot n, \mathtt{Diam}(G) \cdot n)$.

Proof. The bound of the number of paths simply follows by the fact that x is an integral flow of value ℓ, and by the definition of flow decomposition (Def. 1.3). Furthermore as G is a DAG every path of G is of length at most $n-1$. Thus $\mathrm{TotLen}(\mathcal{P}) \leq \ell \cdot n$. To show that $\mathrm{TotLen}(\mathcal{P}) \leq \mathtt{Diam}(G) \cdot n$ we make the following observation. Consider any $P_j \in \mathcal{P}$ and let u and v be two consecutive principal vertices on P_j, in the sense that the only principal vertices on the segment $P_j[u, v]$ are u and v. We then claim that $|P_j[u, v]| \leq \mathtt{Diam}(G)$.

Assume for the sake of contradiction that $|P_j[u, v]| > \mathtt{Diam}(G)$, we will show that in this case, we can replace the path $Q = \mathsf{Translate}^{-1}(P_j)$ with an alternative path $Q^* \subseteq \widetilde{G}$, that provides a strictly lower cost and the same value of flow, hence leading to a contradiction. That is, in the alternative flow solution x' we will add one unit flow over Q^* and omit one unit flow from Q. Letting $R = [u_0 = u, \ldots, u_k = v]$ be the u-v shortest path in G, then the alternative path Q^* is defined as follows:

$$Q^* = Q[., u^{out}] \circ (u^{out}, u_1^{in}) \circ T_1 \circ (u_1^{out}, u_2^{in}) \circ T_2 \circ (u_2^{out}, u_3^{in}) \circ \ldots \circ T_{k-1} \circ (u_{k-1}^{out}, u_k^{inn}) \circ Q[v^{in}, .] \,,$$

where $T_j = (u_j^{in}, u'_j) \circ (u'_j, u_j^{out})$ for every $j \in \{1, \ldots, k-1\}$. Consider the flow solution x' defined by:

$$\begin{cases} x'(e) = x(e) - 1, & \text{for every } e \in Q, \\ x'(e) = x(e) + 1, & \text{for every } e \in Q^*, \\ x'(e) = x(e), & \text{otherwise} \,. \end{cases}$$

We first show that x' is a legal flow, and with the same value as x. This holds as all edges on the paths of T_j for $j \in \{1,\ldots,k-1\}$, have a sufficiently large capacity. Next, we show that $\sum_e c(e)x'(e) < \sum_e c(e)x(e)$, which will be in contradiction to the optimality of x.

By the definition of u and v, $c(Q[u^{out}, v^{in}]) = |P_j[u,v]| - 1$. In the same manner, $c(Q^*[u^{out}, v^{in}]) = |R| - 1$. Since $|R| < |P_j[u,v]|$, we conclude that $c(Q) > c(Q^*)$, and consequently that $\sum_e c(e)x'(e) < \sum_e c(e)x(e)$ as desired. We therefore have that $|P_j[u,v]| \leq \texttt{Diam}(G)$, and $|P_j| \leq \texttt{Diam}(G) \cdot n(P)$. Combining with Obs. 2.2, we have $\sum_{i=1}^{\ell} |P_i| \leq \texttt{Diam}(G) \cdot \sum_{i=1}^{\ell} n(P_i) \leq \texttt{Diam}(G) \cdot n$. ∎

We next show that each long dipath in G is dominated by the vertices of $V(\mathcal{P})$ (i.e., contains a bounded number of uncovered vertices).

Lemma 2.4. *Any dipath in G contains at most $2n/\ell$ vertices of $V \setminus V(\mathcal{P})$.*

Proof. Assume by contradiction that there exists a dipath $Q \subseteq G$ that contains at least $2n/\ell$ vertices of $U = V \setminus V(\mathcal{P})$. Observe that as $|\mathcal{P}| = \ell$, by Obs. 2.2, there exists a path $P^* \in \mathcal{P}$ with $n(P^*) \leq n/\ell$ principal vertices. Therefore $c(\mathsf{Translate}^{-1}(P^*)) \geq -n^3 \cdot n/\ell$ (recall that $c((v_i^{in}, v_i^{out})) = -n^3$ for any $i \in \{1,\ldots,n\}$).

We next show that the path Q implies the existence of an s-t path Q' in $\widetilde{G}$ that has a lower cost than that of $\mathsf{Translate}^{-1}(P^*)$. The path Q' is defined based on $Q = [v_1, v_2, \ldots, v_q]$, as follows:

$$Q' = (s,s') \circ (s', v_1^{in}) \circ T_1 \circ (v_1^{out}, v_2^{in}) \circ T_2 \circ (v_2^{out}, v_3^{in}) \circ \ldots (v_{q-1}^{out}, v_q^{in}) \circ T_q \circ (v_q^{out}, t) \ ,$$

where $T_j = (v_j^{in}, v_j') \circ (v_j', v_j^{out})$ for every $v_j \in V(Q) \setminus U$, and $T_j = (v_j^{in}, v_j^{out}))$ for every $v_j \in V(Q) \cap U$. By the definition of U, $x((v_i^{in}, v_i^{out})) = 0$ for every $v_i \in U$. In addition,

$$c(Q') \leq -n^3 \cdot 2n/\ell + n \leq -n^3 \cdot n/\ell - n^3 + n < c(\mathsf{Translate}^{-1}(P^*)) \ .$$

Consider an alternative flow solution x' obtained by replacing the path P^* with the path Q' in the flow decomposition. That is, let $x'(e) = x(e) - 1$ for every $e \in \mathsf{Translate}^{-1}(P^*)$, and $x'(e) = x(e) + 1$ for every $e \in Q'$. Note that since $x((v_i^{in}, v_i', v_i^{out})) = 0$ for every $v_i \in U$, we have that x' is a legal flow of the same value as that of x, but of a strictly lower cost, as $c(\mathsf{Translate}^{-1}(P^*)) > c(Q')$. Ending with a contradiction to the optimality of x. ∎

Theorem 2.5. *The set of ℓ paths $\mathcal{P}$ maximize the total number of covered vertices (namely, vertices appearing on these paths) and subject to this condition the total path-length* $\mathrm{TotLen}(\mathcal{P})$ *is minimized.*

Proof. Let a be the number of vertices covered by $\mathcal{P}$ and let $b = \mathrm{TotLen}(\mathcal{P})$. Notice that the cost of the output flow of Algorithm PathCover is exactly

$$a \cdot (-n^3) + (b-a) \cdot 1 = a \cdot (-n^3 - 1) + b \ ,$$

which follows from the definition of the edge costs of Algorithm PathCover. Assume towards a contradiction, that there is a valid flow that covers at least $a+1$ vertices, then we will have that the cost of such a flow will be at most

$$(a+1) \cdot (-n^3) + n^2 < a \cdot (-n^3 - 1) \ .$$

Hence, we get a contradiction to the minimum cost property of the output flow. Furthermore, if the flow contains exactly a vertices and the total length of the paths is at most $b-1$, we again get a contradiction to the minimum cost property of the output flow, as

$$a \cdot (-n^3 - 1) + b - 1 < a \cdot (-n^3 - 1) + b \; .$$

This concludes the proof. ■

2.2 Partial ℓ-Covers

The construction of shortcut sets of sublinear size of Theorem 1.5 calls for a variant of the ℓ-covers that we call *partial ℓ-covers*. These covers are defined w.r.t. a given subset V' of vertices that we wish to cover, in the following sense:

Definition 2.2 (Partial ℓ-Covers)**.** Given an n-vertex digraph $G = (V, E)$, a subset $V' \subseteq V$, and a parameter ℓ, a *partial ℓ-cover* for G w.r.t. V' is a multiset of ℓ paths $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ (possibly, singletons or empty), satisfying the following:

1. $\text{TotLen}(\mathcal{P}) \leq 8|V(\mathcal{P}) \cap V'|$,
2. For any long dipath $P \subseteq G$, i.e., such that $|P| \geq 8|V' \cap V(\mathcal{P})|/\ell$, it holds that

$$|V(P) \cap (V' \setminus V(\mathcal{P}))| \leq |V(P)|/4 \; .$$

Theorem 2.6. *For any given n-vertex m-edge DAG $G = (V, E)$, $\ell \in [1, n]$ and $V' \subseteq V$, there is a randomized algorithm* $\mathsf{PartialPathCover}(G, V', \ell)$ *for computing an ℓ-partial cover of G w.r.t. V', w.h.p, in time $\widetilde{O}(m + n^{3/2})$.*

Algorithm PartialPathCover**.** The algorithm is almost equivalent to Algorithm PathCover, up to small adaptations of the min-cost max flow instance. Specifically, we define $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u', c')$, with the same edge and vertex set as in Algorithm PathCover. Letting, u, c be the capacity (resp., cost) functions of Alg. PathCover, then in our reduction, we define: $c'((v_i^{in}, v_i^{out})) = -7$ for every $v_i \in V$, and $c'(e) = c(e)$ for every other edge in $\widetilde{G}$. In addition, $u'((v_i^{in}, v_i^{out})) = 0$ for every $v_i \in V \setminus V'$, and $u'(e) = u(e)$ for every other edge in $\widetilde{G}$.

The remaining algorithm is identical as Algorithm PathCover. It applies Algorithm $\mathsf{MinCostFlow}(\widetilde{G}, s, t)$ using Theorem 1.10. The output flow solution is given by x. Then, it computes the flow-decomposition $\widetilde{\mathcal{P}}$ by applying Lemma 1.11 on x. Finally, the paths $\widetilde{\mathcal{P}}$ are translated into a *multiset* of G-paths $\mathcal{P}$, which provides the output partial ℓ-cover w.r.t. V'. The function Translate provides the bijection from the multiset $\widetilde{\mathcal{P}}$ to the multiset $\mathcal{P}$.

3 Shortcut Algorithms via Path Covers

3.1 Shortcutting to Diameter $D = O(n^{1/2})$

We start by proving Theorem 1.3, by presenting an algorithm for computing linear-size D-shortcuts for $D = O(\sqrt{n})$. This algorithm will be used later on in order to provide improved diameter bounds. By Proposition 1.9, it is sufficient to prove Theorem 1.4 for DAGs. Before

presenting the improved shortcut algorithm FasterShortcutSqrtN, we also need the following definition. For a given directed path $P = [u_1, \ldots, u_k]$ and a vertex v, let u_i be the *first* vertex on P satisfying that $(v, u_i) \in TC(G)$. The edge (v, u_i) is denoted as *the first incoming edge* from v to P, represented by $e(v, P) = (v, u_i)$. Note that the augmented path $P \cup \{e(v, P)\}$ provides a directed path from v to every vertex $u \in P$ such that $(v, u) \in TC(G)$.

Lemma 3.1 (Restatement of Lemma 2 in [KPG+18]). *Let $\mathcal{Q}$ be a collection of directed paths in a DAG. Then, one can compute the edge set $\{e(v, Q) \mid v \in V, Q \in \mathcal{Q}\}$ within $O(|\mathcal{Q}| \cdot m)$ time.*

Algorithm FasterShortcutSqrtN**:**

Input: An n-vertex DAG G,
Output: A shortcut set $H \subseteq TC(G)$ such that $|E(H)| = \widetilde{O}(n)$ and $\mathtt{Diam}(G \cup H) = O(\sqrt{n})$.

1. Compute ℓ-cover $\mathcal{P} = \mathsf{PathCover}(G, \ell)$, for $\ell = \Theta(\sqrt{n})$ (using Theorem 2.2).
2. Let $\mathcal{C} = \mathsf{DisjointChains}(\mathcal{P})$ (using Lemma 1.12).
3. For every $C_i \in \mathcal{C}$, let $H_i = H(C_i)$ be a shortcut set for reducing the diameter of C_i to 2 as obtained by Lemma 1.8.
4. Output $H = \bigcup_{C_i \in \mathcal{C}} C_i \cup H_i$.

Lemma 3.2. *[Runtime and Size] Algorithm* FasterShortcutSqrtN *can be implemented in time $\widetilde{O}(m + n^{3/2})$. In addition, $E(H) = \widetilde{O}(n)$, w.h.p.*

Finally, we complete the diameter argument and establish Theorem 1.3.

Lemma 3.3. *The diameter of $G \cup H$ is at most $O(\sqrt{n})$, w.h.p.*

3.2 Shortcutting to Diameter $D = o(n^{1/2})$

In this subsection, we extend the algorithm of Sec. 3.1 to provide D-shortcuts for $D = o(n^{1/2})$. Obtaining a smaller diameter bound will entitle a cost both in terms of the running time and in the size of the shortcut set. Again, we assume that G is a DAG.

Algorithm FasterShortcutSmallDiam**:**

Input: An n-vertex DAG G and a diameter bound $D = o(n^{1/2})$.
Output: A shortcut set $H \subseteq TC(G)$ satisfying that $\mathtt{Diam}(G \cup H) \leq D$.

1. Let $H_0 \leftarrow \mathsf{FasterShortcutSqrtN}(G)$, (using Theorem 1.3).
2. Compute ℓ-cover $\mathcal{P} = \mathsf{PathCover}(G \cup H_0, \ell)$ for $\ell = 16n/D$, (using Theorem 2.2).
3. Let $\mathcal{C} = \mathsf{DisjointChains}(\mathcal{P})$ (using Lemma 1.12).
4. For every $C_i \in \mathcal{C}$, let $H(C_i)$ be a 2-shortcut set for C_i obtained by Lemma 1.8.
5. Let $V' = V[p]$ and $\mathcal{C}' = \mathcal{C}[p]$ where $p = \Theta(\log n/D)$.

6. Applying Lemma 3.1 to compute $\widehat{H} = \{e(v, C_i) \mid v \in V', C_i \in \mathcal{C}'\}$.

7. Output $H = H_0 \cup \bigcup_{C_i \in \mathcal{C}}(C_i \cup H(C_i)) \cup \widehat{H}$.

Lemma 3.4. *[Runtime and Size] Algorithm* FasterShortcutSmallDiam *can be implemented in time* $\widetilde{O}(m \cdot n/D^2 + n^{3/2})$. *In addition,* $E(H) = \widetilde{O}(n^2/D^3)$, *w.h.p.*

Lemma 3.5 (Diameter Bound)**.** $\texttt{Diam}(G \cup H) \leq D$, *w.h.p.*

Consider a u-v shortest path P in $G' = G \cup H_0 \cup \bigcup(C_i \cup H(C_i))$. Let P', P'' be the $(D/4)$-length prefix (resp., suffix) of P.

Lemma 3.6. *P'' contains at most $D/8$ vertices from $V \setminus V(\mathcal{C})$.*

By the exact same argument as in the proof of Lemma 3.3, we also have:

Observation 3.1. $|V(P'') \cap V(C_i)| \leq 3$ for every $C_i \in \mathcal{C}$.

Therefore by Lemma 3.6, P'' contains representatives vertices from $\Omega(D)$ distinct paths in $\mathcal{C}$. That is, $|\{C_i \in \mathcal{C} \mid V(C_i) \cap V(P'') \neq \emptyset\}| = \Omega(D)$. As each chain in $\mathcal{C}$ is sampled into $\mathcal{C}'$ independently with probability p, by the Chernoff bound, we have that w.h.p., P'' contains at least one vertex $y \in C_i$ for some sampled chain $C_i = [a_1, \ldots, a_k] \in \mathcal{C}'$. In addition, w.h.p., the prefix P' contains at least one sampled vertex $x \in V'$. We then have that the edge $e(x, C_i)$ is in $\widehat{H} \subseteq H$. Let $e(x, C_i) = (x, z)$ for $z \in C_i[a_1, y]$. Therefore the augmented graph $G \cup H$ contains a u-v path $Q = P[u, x] \circ (x, z) \circ C_i[z, y] \circ P[y, v]$ where

$$\begin{aligned} \text{dist}_{G \cup H}(u, v) &\leq |Q| = |P[u, x]| + 1 + |C_i[z, y]| + |P[y, v]| \\ &\leq |P'| + 1 + \text{dist}_{C_i \cup H(C_i)}(z, y) + |P''| \\ &\leq D/4 + 1 + 2 + D/4 \leq D/4 + D/4 + 3 \leq D\,. \end{aligned}$$

This concludes the proof of Theorem 1.4.

3.3 Shortcut Algorithms for Diameter $D = \Omega(n^{1/3})$

In this section, we provide a proof for Theorem 1.5 and compute D-shortcuts of sublinear size for any input $D = \omega(n^{1/3})$. We start by showing that in this case, as well, one can assume that the input graph G in Theorem 1.5 is a DAG. The standard reduction to a DAG is based on adding $\Theta(n)$ shortcut edges, thus too costly for our setting. Then, we present an algorithm that given a collection of dipaths $\mathcal{P}$ computes a D'-shortcut for each path such that the total size of the union of shortcuts in *sublinear*. Again, as we aimed at obtaining sublinear shortcuts we cannot simply use Lemma 1.8 as is. Finally, we present Alg. FasterShortcutLargeD that computes the shortcuts given these tools.

Tool 1: Reduction to DAGs with Sublinear Size

In this subsection, we show:

Theorem 3.7. *Given a digraph $G = (V, E)$, one can compute in time $O(|V| + |E|)$ a DAG $G' = (V', E')$ such that $|V'| = O(|V|)$ and $|E'| = O(|E|)$, and in addition the following holds: Given any D-shortcut set $H' \subseteq TC(G')$ for G', one can compute in time $O(|H'|)$ a D-shortcut set $H \subseteq TC(G)$ for G such that $|H| \leq |H'|$.*

We define a transformed DAG graph G' on which the shortcut computation will be employed, as follows. Denote $|V| = n$ and $|E| = m$, and let $V = \{v_1, \ldots, v_n\}$. We assume w.l.o.g. that G is weakly connected, hence $m \geq n - 1$. Let $C_1, \ldots, C_t$ be the strongly connected components of G. For ease of notation, assume that $v_i \in C_i$ for every $1 \leq i \leq t$. For each component C_i let $\texttt{BFS}_{in}(v_i)$ (resp., $\texttt{BFS}_{out}(v_i)$) be the incoming (resp., outgoing) BFS tree rooted at v_i. Since C_i is strongly connected, each of these trees span all vertices in C_i. For a vertex v_i, let $C(v_i)$ be the strongly connected component of v_i.

The DAG $G' = (V', E')$ is defined as follows. The vertex set V' consists of $2n$ vertices, where for each vertex $v_i \in G$, we include in V' two copies v_i^{in} and v_i^{out}. Hence, $V' = \{v_i^{in}, v_i^{out} \mid v_i \in V\}$. The edge set E' is the union of the following subsets:

1. $E^{in} = \{(v_j^{in}, v_k^{in}) \mid (v_j, v_k) \in \texttt{BFS}_{in}(v_i), i \in \{1, \ldots, t\}\}$.
2. $E^{out} = \{(v_j^{out}, v_k^{out}) \mid (v_j, v_k) \in \texttt{BFS}_{out}(v_i), i \in \{1, \ldots, t\}\}$.
3. $\widehat{E} = \{(v_i^{out}, v_j^{in}) \mid (v_i, v_j) \in E \text{ and } C(v_i) \neq C(v_j)\}$
4. $E^{bridge} = \{(v_i^{in}, v_i^{out}) \mid i \in \{1, \ldots, t\}\}$.

Then, $E' = E^{in} \cup E^{out} \cup \widehat{E} \cup E^{bridge}$, see[12] Fig. 2 for an illustration. Notice that $|E'| \leq m + 3n = O(m)$.

We say that a vertex v_i in G *corresponds* to its two copies vertices v_i^{in} and v_i^{out} in G' and vice-versa. Furthermore an edge (v_i, v_j) with $i \neq j$ in graph G *corresponds* to all existing edges in G' of the form

$$(v_i^{in}, v_j^{in}), (v_i^{in}, v_j^{out}), (v_i^{out}, v_j^{in}), (v_i^{out}, v_j^{out}) \tag{2}$$

and vice versa. For every edge $e' \in \{(v_i^{in}, v_j^{in}), (v_i^{in}, v_j^{out}), (v_i^{out}, v_j^{in}), (v_i^{out}, v_j^{out})\}$, let $g(e') = (v_i, v_j)$. For every $C_i \subseteq G$, let $C_i' = \{v_j^{in}, v_j^{out}, \mid v_j \in C_i\}$ be the corresponding component in G'. Let $T_i = G'[C_i']$ be the induced graph of G' on the vertex set C_i'. Therefore, $E(T_i) = \texttt{BFS}_{in}(v_i) \cup \texttt{BFS}_{out}(v_i) \cup \{(v_i^{in}, v_i^{out})\}$. We start by observing that the transformed graph is a DAG.

Observation 3.2. G' is a DAG.

Proof. This follows from the following observations:

1. For each $1 \leq i \leq t$, $T_i \subseteq G'$ is a DAG.
2. The graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ obtained by contracting each strongly connected component C_i into a super-node $\widetilde{v}_i$ (omitting self-loops) is a DAG. Formally, $\widetilde{V} = \{\widetilde{v}_i, i \in \{1, \ldots, t\}\}$ and $\widetilde{E} = \{(\widetilde{v}_i, \widetilde{v}_j) \mid \exists v_a \in C_i, v_b \in C_j \text{ such that } (v_a, v_b) \in G \text{ and } i \neq j\}$.

[12] Note that for (v_i, v_j) edges where $C(i) \neq C(j)$, we add edges (v_i^{out}, v_j^{in}). In addition, we have in E' edges of the form (v_i^{in}, v_i^{out}) (getting into the out-copy of v_i).

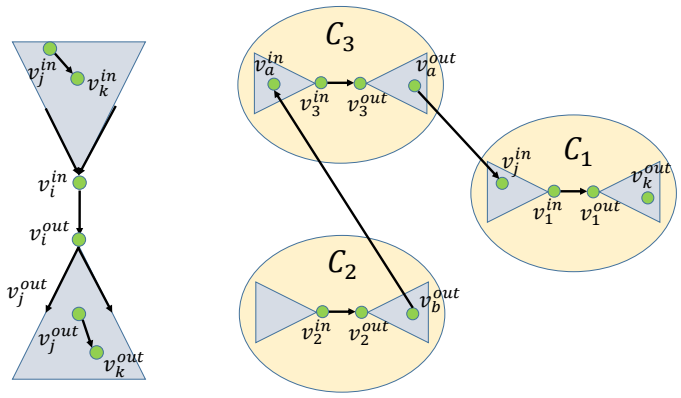

Figure 2: An illustration for the DAG reduction of Theorem 3.7. Left: Illustration for a single strongly connected component C_i. The incoming BFS tree (resp., outgoing BFS) rooted at v_i spans all the in-copy (resp., out-copy) of the vertices in C_i. Right: Illustration of three strongly connected components. A G-edge (v_b, v_a) for $v_b \in C_2$ and $v_a \in C_3$ corresponds to the edge (v_b^{out}, v_a^{in}) in G'. Similarity the G-edge (v_a, v_j) for $v_j \in C_1$ corresponds to the edge (v_a^{out}, v_j^{in}) in G'.

3. The graph $\widetilde{G}' = (\widetilde{V}', \widetilde{E}')$ obtained by contracting all edges in T_i, for every $i \in \{1, \ldots, t\}$, is a DAG. The vertex set $\widetilde{V}' = \{\widetilde{v}'_i \mid i \in \{1, \ldots, t\}\}$.

To conclude the proof that G' is a DAG assume towards contradiction that it contains a directed cycle Q. From (1) it follows that this cycle must contain vertices from at least two distinct subgraphs T_i and T_j, Therefore, the contracted graph $\widetilde{G}'$ contains a directed cycle as well, contradiction to (3). ■

Observation 3.3. (i) For every $(v_i, v_j) \in TC(G)$, it holds that $(v_i^{in}, v_j^{out}) \in TC(G')$, and (ii) for every $(u, v) \in TC(G')$, it holds that $g(u, v) \in TC(G)$.

Proof. (i) Assume first that $C(v_i) = C(v_j) = C_x$. Then T_x contains the dipath

$$\pi(v_i^{in}, v_x^{in}, \texttt{BFS}_{in}(v_x)) \circ (v_x^{in}, v_x^{out}) \circ \pi(v_i^{out}, v_j^{out}, \texttt{BFS}_{out}(v_x)) \ ,$$

where $\pi(a, b, T)$ is the tree path from a to b in the directed tree T.

Next, assume that $C(v_i) \neq C(v_j)$ where $C_x = C(v_i)$ and $C_y = C(v_j)$. Let P' be the $C_x \rightsquigarrow C_y$ path in $\widetilde{G}$, where $P' = [C_x = C_{j,1}, C_{j,2}, \ldots, C_y = C_{j,k}]$. For every $q \in \{1, \ldots, k-1\}$, let $e_q = (v_{j,q}, v_{j,q+1})$ be a G-edge such that $v_{j,q} \in C_{j,q}, v_{j,q+1} \in C_{j,q+1}$. We then have that $(v_{j,q}^{out}, v_{j,q+1}^{in}) \in \widehat{E}$ for $q \in \{1, \ldots, k-1\}$. Therefore, there is a G'-dipath from v_i^{in} to v_j^{out} given by:

$$\pi(v_i^{in}, v_{j,1}^{out}, T_{j,1}) \circ (v_{j,1}^{out}, v_{j,2}^{in}) \circ \ldots \circ (v_{j,k-1}^{out}, v_{j,k}^{in}) \circ \pi(v_{j,k}^{in}, v_{j,k}^{out}, T_{j,k}) \ ,$$

where for every C_x and every $v_a, v_b \in C_x$, $\pi(v_a^{in}, v_b^{out}, T_x)$ denote the dipath $v_a^{in} \rightsquigarrow v_b^{out}$ in T_x. ■

A D-Shortcut H' for G' Yields a D-Shortcut H for G. We start by observing that for any edge $(u,v) \in TC(G')$, it holds that $g(u,v) \in TC(G)$. Let $g(u,v) = (v_i, v_j)$ then the interesting case is when $i \neq j$. First assume that $C(v_i) = C(v_j)$, then since $C(v_i)$ is strongly connected in G, we have that $(v_i, v_j) \in TC(G)$. Next, assume that $C(v_i) \neq C(v_j)$. Let $C_x = C(v_i)$ and $C_y = C(v_j)$. Since $(u,v) \in TC(G')$, it implies that there is $\widetilde{v}'_x \rightsquigarrow \widetilde{v}'_y$ dipath in $\widetilde{G}'$. Since every edge $(\widetilde{v}'_a, \widetilde{v}'_b)$ corresponds to an G-edge $(v_{i'}, v_{j'})$ for some $v_{i'} \in C_a$ and $v_{j'} \in C_b$, we have that there is an $\widetilde{v}_x \rightsquigarrow \widetilde{v}_y$ dipath in $\widetilde{G}$. As C_x, C_y are strongly connected components, it holds that $(v_i, v_j) \in TC(G)$.

Let $H' \subseteq TC(G')$ be a D-shortcut for G'. We claim that $H = \{g(u,v) \mid (u,v) \in H'\}$ is a D-shortcut for G. First, by the observation above, we have that $H \subseteq TC(G)$. It remains to bound the diameter of $G \cup H$. Fix a pair $v_i, v_j \in V(G)$ and let P be a v_i-v_j shortest path in $G \cup H$. First assume that $C(v_i) \neq C(v_j)$. By construction of G', we have that $(v_i^{out}, v_j^{in}) \in TC(G')$. Let P' be a (v_i^{out}, v_j^{in}) shortest path in G'. Now the proof follows as $|P| \leq |P'|$. This follows since we can build a v_i-v_j walk W of length at most D in G by doing the following. For each $e \in P'$ we add edge $g(e)$ to W. hence $|P| \leq |W| \leq |P'| \leq D$. The proof when $C(v_i) = C(v_j)$ is almost identical as thus omitted.

Running time. The computation of the strongly connected components of G can be done in linear time [Tar72]. It is easy to see that $|E'| = O(m)$ and $|V'| = O(n)$. Given H' computing the corresponding set $H = \{g(u,v) \mid (u,v) \in G\}$ can be done in linear time as well. This completes the proof of Theorem 3.7.

Tool 2: Shortcutting Dipaths with Sublinear Number of Edges. A crucial tool in our prior constructions is based on computing 2-shortcuts of nearly linear size for dipaths. For our purposes, we next provide an alternative scheme that provides shortcuts of sublinear size at the cost of increasing the diameter.

Lemma 3.8. *[Shortcuts of Dipaths with Sublinear Size] There exists an algorithm* PathShortcut *that given a DAG G, dipath collection $\mathcal{P}$ and integer parameter d, outputs $H \subseteq TC(G)$ satisfying that for all $1 \leq i \leq t$, we have $\texttt{Diam}(P_i \cup H[V(P_i)]) = O(d)$ and $|H| = \widetilde{O}(\sum_{j=1}^{t} |P_i|/d)$. The running time is $\widetilde{O}(\sum_{P \in \mathcal{P}} |V(P|)$.*

The Shortcut Algorithm. We are now ready to present Algorithm FasterShortcutLargeD. The algorithm is given as input a DAG G and diameter bound $D = \omega(n^{1/3})$. Define:

$$t_D = \sqrt{n/D} \quad \text{and} \quad s_D = D^{3/2}/\sqrt{n}\,. \tag{3}$$

The algorithm consist of $i^* \leq t_D$ iterations (as will be shown in the analysis), where in each iteration i, the algorithm computes a partial t_D-cover $\mathcal{P}_i$ with respect to the set $V_i = V \setminus \bigcup_{j=0}^{i-1} \mathcal{P}_i$, namely, the set of vertices that are not yet covered by the current collection of paths $\bigcup_{j=0}^{i-1} \mathcal{P}_i$. It then computes an s_D-shortcut $H(P)$ for each $P \in \mathcal{P}_i$ by applying Algorithm PathShortcut of Lemma 3.8, and adds $\bigcup_{P \in \mathcal{P}_i} H(P)$ to the output shortcut H.

At the end of these iterations, we obtain $i^* \leq t_D$ path collections $\mathcal{P}_0, \ldots, \mathcal{P}_{i^*-1}$, where $|\mathcal{P}_{j-1}| \leq t_D$ for every $j \in \{1, \ldots, i^*\}$. The algorithm then defines a collection of $\Theta(t_D \log n)$ paths $\mathcal{R}$ as follows. The dipaths of the last path collection, namely, $\mathcal{P}_{i^*-1}$ is taken *entirely* into $\mathcal{R}$. In addition, each other dipath in $\bigcup_{j=1}^{i^*-2} \mathcal{P}_j$ is sampled into $\mathcal{R}$ independently with probability $p = \Theta(\log n / t_D)$.

By the Chernoff bound we get that, w.h.p., $|\mathcal{R}| = \Theta(t_D \log n)$ as desired. The algorithm then adds to H, the collection of $e(v, P)$ edges for every $v, P \in V[p'] \times \mathcal{R}$ where $p' = \Theta(\log n / D)$. This completes the description of the algorithm.

Algorithm FasterShortcutLargeD**:**

Input: An n-vertex DAG $G = (V, E)$ and a diameter bound $D = \omega(n^{1/3})$.
Output: $H \subseteq TC(G)$ satisfying that $\texttt{Diam}(G \cup H) = O(D)$ and $|H| = \widetilde{O}((n/D)^{3/2})$.

1. Set $V_0 = V$, $U_0 = V$, $i = 0$, $p' = 10 \log n \cdot D^{-1}$ and $p = 10 \log n \cdot t_D^{-1}$.
2. While $|U_i| \geq \frac{n}{t_D}$ do the following:
   (a) Set $\mathcal{P}_i = \mathsf{PartialPathCover}(G, V_i, t_D)$.
   (b) Set $H_i = \mathsf{PathShortcut}(\mathcal{P}_i, s_D)$.
   (c) $U_{i+1} \leftarrow V(\mathcal{P}_i) \cap V_i$, $V_{i+1} \leftarrow V_i \setminus V(\mathcal{P}_i)$, and $i \leftarrow i + 1$.
3. Set $\mathcal{R} = \bigcup_{j=0}^{i-2} \mathcal{P}_j$, $\mathcal{R}' = \mathcal{R}[p] \cup \mathcal{P}_{i-1}$ and $V' = V[p']$.
4. $\widehat{H} = \{e(v, P_i) \mid v \in V', P_i \in \mathcal{R}'\}$ (using Lemma 3.1).
5. Output $H = \bigcup_{j=0}^{i-1} H_i \bigcup \widehat{H}$.

Analysis. Recall that the index i^* indicates the largest index of the application, where $|U_{i^*}| < n/t_D$ leading to the termination of the algorithm.

Observation 3.4. $V = \bigcup_{j=0}^{i^*-1} V(\mathcal{P}_j) \cup V_{i^*}$.

Observation 3.5. For every $Q \in \bigcup_{j=0}^{i^*-1} \mathcal{P}_j$, for every u-v shortest path $P \subseteq G \cup \bigcup_{j=0}^{i^*-1} H_j$ it holds that $|V(Q) \cap V(P)| = O(s_D)$.

Lemma 3.9. *[Runtime and Size] Algorithm* FasterShortcutLargeD *can be implemented in time* $\widetilde{O}((m + n^{3/2}) \cdot t_D)$. *In addition,* $|E(H)| = \widetilde{O}((n/D)^{3/2})$, *w.h.p.*

Proof. We first claim that the While loop is applied at most t_D times. By definition, $U_j \cap V_j = \emptyset$ and $U_j \cup V_j = V_{j-1}$. In addition, since $U_j \subseteq V_{j-1}$, it holds that $U_1, U_2, \ldots, U_i$ are vertex-disjoint. Since the While loop continues only provided that $|U_i| \geq n/t_D$, there can be at most t_D applications of this loop. We will need the following observation.

Observation 3.6. $\text{TotLen}(\mathcal{R} \cup \mathcal{P}_{i^*-1}) = O(n)$.

Runtime: We focus on a single application of the While loop and show that it can be implemented in $\widetilde{O}(m + n^{3/2})$ time, since there are at most t_D applications, the final runtime is bounded by $\widetilde{O}((m + n^{3/2})t_D)$. Step (a) is implemented in $\widetilde{O}(m + n^{3/2})$ time by Theorem 2.6. Step (b) runs in $\widetilde{O}(\text{TotLen}(\mathcal{R} \cup \mathcal{P}_{i-1})) = \widetilde{O}(n)$, by Lemma 3.8 and Observation 3.6 (we note that in fact this is the total running time of Step (b) over all the iterations of the While loop). Step (c) runs in $O(n)$ time. The runtime bound of the t_D applications of the While loop follows. We next analyze the remaining steps of the algorithm. By Observation 3.6, Step 3 is implemented in $O(n)$ time. Since

$|\mathcal{R}'| = |\mathcal{R}[p]| + |\mathcal{P}_{i-1}|$, w.h.p., we have that $|\mathcal{R}'| = \widetilde{O}(t_D)$ as $|\mathcal{R}[p]| = \widetilde{O}(t_D)$ (this follows from the fact that $|\mathcal{R}| \leq t_D^2$ as there are at most t_D iterations of line 2(a) and at each iterations at most t_D paths are created) and $|\mathcal{P}_{i-1}| \leq t_D$. Using Lemma 3.1, Step 4 is implemented in $\widetilde{O}(m \cdot t_D)$.

Size: By the Chernoff bound, w.h.p., it holds that $|\widehat{H}| = |V'| \cdot |\mathcal{R}'| = |V'| \cdot (|\mathcal{R}[p]| + |\mathcal{P}_{i^*-1}|) = \widetilde{O}\left(\frac{n}{D} \cdot (t_D + t_D)\right) = \widetilde{O}((n/D)^{3/2})$. Furthermore, we also have $\sum_{j=0}^{i^*-1} |H_j| = O((n/D)^{3/2})$. This follows from the fact proven in Observation 3.6 that $\sum_{P \in \mathcal{R} \cup \mathcal{P}_{i^*-1}} |V(P)| = O(n)$. Hence, by Lemma 3.8 and Step 2(b) of the algorithm, w.h.p., we have that

$$\sum_{j=0}^{i^*-1} |H_j| = \widetilde{O}\left(s_D^{-1} \cdot \sum_{P \in \mathcal{R} \cup \mathcal{P}_{i-1}} |V(P)|\right) = \widetilde{O}(s_D^{-1} \cdot n) = \widetilde{O}((n/D)^{3/2}).$$

We conclude that $|H| = |\bigcup_{j=0}^{i^*-1} H_j \bigcup \widehat{H}| = \widetilde{O}((n/D)^{3/2})$, w.h.p. ■

Lemma 3.10 (Diameter Bound). $\texttt{Diam}(G \cup H) = O(D)$, *w.h.p.*

Proof. Consider a u-v shortest dipath P in the graph $G' = \bigcup_{j=0}^{i^*-1} H_j \cup G$. Let P', P'' be the $8D$-length prefix and suffix of P, respectively. By the Chernoff bound, it holds that $V(P') \cap V' \neq \emptyset$, and let u^* be some arbitrary vertex in $V(P') \cap V'$. Recall that $\mathcal{P}_{i^*-1}$ is the last path collection obtained by applying Alg. PartialPathCover We next consider two cases.

Case 1: $V(P'') \cap V(\mathcal{P}_{i^*-1}) \neq \emptyset$**.** Let v^* be some vertex in $V(P'') \cap V(\mathcal{P}_{i^*-1})$ and let $Q \in \mathcal{P}_{i^*-1}$ be some path satisfying that $v^* \in V(Q)$. Since $Q \in \mathcal{R}'$, the edge $e(u^*, Q)$ is in $\widehat{H}$, and thus also in the output shortcut H. Let $e(u^*, Q) = (u^*, z)$ where z is the first vertex on Q such that there is a dipath from u^* to z in G. We therefore have that:

$$\begin{aligned} \text{dist}_{G \cup H}(u, v) &\leq \text{dist}_{G'}(u, u^*) + \text{dist}_{G' \cup \widehat{H}}(u^*, z) + \text{dist}_{G \cup H_{i^*-1}}(z, v^*) + \text{dist}_{G'}(v^*, v) && (4) \\ &\leq |P'| + 1 + O(s_D) + |P''| = O(D)\ . && (5) \end{aligned}$$

This holds as $\text{dist}_{G \cup H_{i-1}}(z, v^*) = O(s_D)$ by Lemma 3.8 and $s_D = D \cdot \sqrt{D/n} \leq D$.

Case 2: $V(P'') \cap V(\mathcal{P}_{i^*-1}) = \emptyset$**.** We will need the following observations.

- Observation 1: $V(P'') \subseteq \bigcup_{j=0}^{i^*-2} \mathcal{P}_j \cup V_{i^*}$. This observation follows from Obs. 3.4, as $V(P'') \cap V(\mathcal{P}_{i^*-1}) = \emptyset$.
- Observation 2: $8|U_{i^*}|/t_D \leq 8D$. This observation follows from the fact that $|U_{i^*}| \leq n/t_D$.

Hence by property (2) of Definition 2.2 of the partial ℓ-cover $\mathcal{P}_{i^*-1}$ w.r.t. V_{i^*-1} (with $\ell = t_D$), we have that:

$$\begin{aligned} |V(P'') \cap V_{i^*-1}| &= |V(P'') \cap V_{i^*}| && \text{By Observation 1.} \\ &\leq |V(P'')|/4 && \text{By property (2) of Definition 2.2 and Observation 2.} \end{aligned}$$

By Obs. 3.5, it holds that $|V(Q) \cap V(P'')| = O(s_D)$ for every $Q \in \bigcup_{j=0}^{i^*-2} \mathcal{P}_j$. Therefore, P'' must contain vertices from at least $\Omega(D/s_D) = \Omega(t_D)$ distinct paths in $\mathcal{P}' = \bigcup_{j=0}^{i^*-2} \mathcal{P}_j$. Formally, let $\mathcal{X} = \{Q \in \mathcal{P}' \mid V(Q) \cap V(P'') \neq \emptyset\}$, then $|\mathcal{X}| = \Omega(t_D)$.

Since each path in $\mathcal{X}$ is sampled into $\mathcal{R}'$ independently with probability of $p = \Theta(\log n \cdot t_D^{-1})$, by the Chernoff bound, w.h.p., $|\mathcal{X} \cap \mathcal{R}'| \neq \emptyset$. Letting $Q^* \in \mathcal{X} \cap \mathcal{R}'$, we have that there is a vertex $y \in V(Q^*) \cap V(P'')$ and in addition, the edge $e(u^*, Q^*) = (u^*, z)$ is in H, where z is the first vertex on Q^* from which there is an incoming path from u^*. Altogether by the same argument as in Case 1, we have that $\text{dist}_{G \cup H}(u, v) = O(D)$. ■

4 Applications

4.1 Minimum Chain Covers and Maximum Antichains

Recall that a chain is dipath in $TC(G)$ and an antichain is a subset of vertices that are independent set in $TC(G)$. In this section, we prove Theorem 1.6. For a dipath collection $\mathcal{P}$, let $\text{TotLen}(\mathcal{P}) = \sum_{P\in\mathcal{P}} |P|$. Note that it might be the case that $\text{TotLen}(\mathcal{P}) \gg |V(\mathcal{P})|$ (in case where the dipaths are not vertex-disjoint, and each vertex appears on multiple paths).

We need the following definitions.

Definition 4.1 (Minimum Path Covers)**.** A minimum path cover (MPC) of a DAG $G = (V, E)$ is a minimum-size set of dipaths in G that covers V.

Lemma 4.1. *Given an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ for a DAG $G = (V, E)$, the chain collection $\mathcal{C} = \text{DisjointChains}(\mathcal{P})$ is an MCC. Given MPC, the MCC can be thus computed in time $O(\text{TotLen}(\mathcal{P}))$.*

The MCC algorithm is based on combining the following three ingredients:

1. Computing an MPC $\mathcal{P}$ (consisting of G-dipaths) in time $\widetilde{O}(m + n^{3/2} + \text{TotLen}(\mathcal{P}))$.
2. Showing that every digraph G has an MPC $\mathcal{P}$ of $\text{TotLen}(\mathcal{P}) \leq n \cdot \texttt{Diam}(G)$.
3. Computing a *Minimum-Length* MPC $\mathcal{P}$ in time $\widetilde{O}(m + n^{3/2} + \text{TotLen}(\mathcal{P}))$.

In Sec. B.1.1 we introduce the key tool of computing MPC of minimum length, and then in Sec. B.1.2 we provide the detailed MCC algorithm.

4.1.1 Minimum Length Cover

We start by introducing the notion of minimum-length covers and then show how to compute them as a function of their length.

Definition 4.2 (Minimum Length Covers)**.** For a DAG G and input parameter $\ell \geq \omega(G)$, an ℓ-Minimum Length Cover (MLC) is a G-dipath collection $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ satisfying that $V(\mathcal{P}) = V$ (i.e., $\mathcal{P}$ is a path-cover). In addition, $\text{TotLen}(\mathcal{P})$ is the minimum over any other path cover of cardinality ℓ. That is, for any other collection of G-dipaths $\mathcal{P}'$ that covers V and $|\mathcal{P}'| = \ell$, it holds that $\text{TotLen}(\mathcal{P}') \geq \text{TotLen}(\mathcal{P})$.

We next present an algorithm that given a DAG G and a parameter ℓ, either reports that $\omega(G) > \ell$, or else outputs an ℓ-MLC. Algorithm MinimumLengthCover starts by computing the min-cost max-flow $x \in \mathbb{R}^{|\widetilde{E}|}$ of the graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$ (defined as in Alg. PathCover). The case where $\ell < \omega(G)$ is detected by the algorithm based on the cost of the flow. Specifically, in the case where $c(x) = \sum_{e\in\widetilde{E}} c(e)x(e) \geq n^3 - n^4$, the algorithm determines that $\ell < \omega(G)$ and aborts. Intuitively, since the edges (v_i^{out}, v_i^{in}) have negative cost of $-n^3$, in case where the flow solution x *covers* all vertices, it should have a sufficiently small cost. Alternatively, in the case where $\ell \geq \omega(G)$, the algorithm proceeds as follows. It computes the flow-decomposition of x, which we denote by $\mathcal{P}$. The output ℓ-MLC is given by the translated G-paths that correspond to $\mathcal{P}$. This completes the description of the algorithm.

Algorithm MinimumLengthCover

Input: An n-vertex DAG G and $\ell \in \mathbb{N}_{\geq 1}$.
Output: For $\ell < \omega(G)$, return -1. Otherwise, return an ℓ-MLC.

1. Let $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$ be defined as in Alg. PathCover (Theorem 2.2).
2. Apply Algorithm MinCostFlow$(\widetilde{G}, s, t)$ (Theorem 1.10), and let $x \in \mathbb{R}^{|\widetilde{E}|}$ be the output flow.
3. If $c(x) \geq n^3 - n^4$, return -1.
4. **Flow Decomposition.** Decompose x into a multiset of s-t paths $\mathcal{P}'$ in $\widetilde{G}$ by applying Lemma 1.11.
5. Return $\mathcal{P} = \{\mathsf{Translate}(P') \mid P' \in \mathcal{P}'\}$.

Correctness. The minimality of the cover follows by Theorem 2.5. Let ℓ be the input parameter to MinimumLengthCover, and recall $c(x) = \sum_{e \in \widetilde{E}} x(e) \cdot c(e)$ be the cost of the output flow $x = \mathsf{MinCostFlow}(\widetilde{G}, s, t)$.

Observation 4.1. If $\omega(G) \leq \ell$, then $c(x) \leq n^2 - n^4$, and otherwise $c(x) \geq n^3 - n^4$.

Proof. Assume that $\omega(G) \leq \ell$. Observe that in this case $x(v_i^{in}, v_i^{out}) = 1$ for every $i \in \{1, \ldots, n\}$. Since the value of the flow is ℓ and $c(P') \leq n$ for every dipath $P' \subseteq \widetilde{G}$, we have that $c(x) \leq \sum_{i=1}^{n} c(v_i^{in}, v_i^{out}) \cdot x(v_i^{in}, v_i^{out}) + n^2 \leq n^2 - n^4$.

In the case where $\omega(G) > \ell$, there is at least one vertex v_i that is uncovered by the paths $\mathcal{P}$. This implies that $x(v_i^{in}, v_i^{out}) = 0$, and therefore we have that $c(x) \geq -(n-1)n^3$. ■

Claim 4.2. *If $\ell \geq \omega(G)$, then* $\mathrm{TotLen}(\mathcal{P}) = c(x) + n^4 + n$.

Proof. This follows from the fact that there are n edges of the form (v_i^{in}, v_i^{out}) and $\sum_{i=1}^{n} c(v_i^{in}, v_i^{out}) \cdot x(v_i^{in}, v_i^{out}) = -n^4$ and the rest of the edges that contribute to M are of the form (v_i^{in}, v_i') for which $c(v_i^{in}, v_i') = 1$. That is, we have the following identity:

$$\sum_{i=1}^{n} c(v_i^{in}, v_i') \cdot x(v_i^{in}, v_i') + n = \sum_{i=1}^{n} x(v_i^{in}, v_i') + n = \sum_{i=1}^{k} |P_i| \,.$$

The claim follows by the optimality of MinCostFlow (i.e., that x is of minimum-cost). ■

The running time is $\tilde{O}(m + n^{3/2} + \mathrm{TotLen}(\mathcal{P}))$ by Theorem 1.10.

4.1.2 Algorithm for Computing Minimum Chain Covers

We are now ready to describe the MCC algorithm that establishes Theorem 1.6. The algorithm starts by reducing the diameter of G to $\sqrt{n}$ by computing a $\sqrt{n}$-shortcut set H_0 (using Theorem 1.3). It then computes an $\omega(G)$-MLC $\mathcal{P}$. Note that the dipaths of $\mathcal{P}$ are in $G \cup H_0$ and are not

necessarily vertex-disjoint. Finally, the algorithm transforms the $\omega(G)$-MLC into an MCC using Alg. DisjointChains. Since the complexity of the latter algorithm depends on $\text{TotLen}(\mathcal{P})$, the runtime argument will (i) exploit the fact that $\mathcal{P}$ has minimum total-length, and (ii) provide an upper on the total length of an ℓ-MLC.

Algorithm MinimumChainCover**:**

Input: An n-vertex DAG G with m edges.
Output: An MCC of G.

1. Let $H_0 \leftarrow \text{FasterShortcutSqrtN}(G)$ where $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$ (using Theorem 1.3), and set $G' = G \cup H_0$.
2. Compute $\omega(G')$ by Binary search by applying Alg. $\text{MinimumLengthCover}(G', \ell)$ (where ℓ is the width guess).
3. $\mathcal{P} = \text{MinimumLengthCover}(G', \omega(G'))$.
4. Output $\mathcal{C} = \text{DisjointChains}(\mathcal{P})$

Analysis. Observe that $\omega(G) = \omega(G')$ as $H_0 \subseteq TC(G)$. Since an $\omega(G)$-Minimum-Length-Cover $\mathcal{P}$ is a path cover with cardinality $\omega(G)$, we have that:

Claim 4.3 (Correctness)**.** *$\mathcal{C}$ is an MCC, w.h.p.*

To bound the running time, we need the following lemma.

Lemma 4.4. *Any DAG G has an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of total length* $\text{TotLen}(\mathcal{P}) \leq n \cdot \texttt{Diam}(G)$.

Proof. Let $\mathcal{C}$ be some MCC of G (see Lemma B.1). Since the chains in $\mathcal{C}$ are vertex-disjoint, we have that $\sum_{C_i \in \mathcal{C}} |C_i| \leq n$. We transform each chain C_i into a path $P_i \subseteq G$. This can be done by traversing C_i and replacing each $(u, v) \in C_i \setminus G$ by the u-v shortest path $P_{u,v}$ in G. Formally, letting $C_i = [u_1, \ldots, u_q]$ then $P_i = P_{u_1,u_2} \circ P_{u_2,u_3} \circ \ldots \circ P_{u_{q-1},u_q}$. Since G is a DAG, P_i is a dipath in G. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ where $k = \omega(G)$.

As all chains in $\mathcal{C}$ are vertex-disjoint and covering V, we have that $\mathcal{P}$ is an MPC (as each path P_i is contained in G). In addition, since each edge (u, v) in C_i is replaced in P_i by a u-v shortest path in G, we have that $|P_i| \leq |C_i| \cdot \texttt{Diam}(G)$, as . Therefore,

$$\sum_{i=1}^{k} |P_i| \leq \texttt{Diam}(G) \cdot \sum_{i=1}^{k} |C_i| \leq \texttt{Diam}(G) \cdot n.$$

■

Claim 4.5. *Algorithm* MinimumChainCover *can be implemented in time $\widetilde{O}(m + n^{3/2})$.*

Proof. The computation of the shortcut H_0 takes $\widetilde{O}(m + n^{3/2})$ time by Theorem 1.3. Each application of Alg. $\text{MinimumLengthCover}(G', \ell)$ takes $\widetilde{O}(m + n^{3/2})$ (by Lemma B.4 and the fact that $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$) and there are $O(\log n)$ such applications. By the correction of Alg. MinimumLengthCover, once again we have by Lemma B.4 that $\sum_{i=1}^{k} |P_i| \leq \texttt{Diam}(G \cup H_0) \cdot n = O(n^{3/2})$. Therefore, the computation of $\mathcal{C}$ takes $O(n^{3/2})$ time. ■

4.1.3 Maximum Antichains

Recall that an antichain of a DAG G is a set of pairwise non-reachable vertices in G. Our goal is to compute an antichain of maximum size. The next lemma follows immediately from the proof of Lemma 2 in [CCM+21b].

Lemma 4.6. *[Restatement of Lemma 2 in [CCM+21b]] Given a DAG $G = (V, E)$ and an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_\ell\}$, one can compute a maximum antichain of G in time $O(\sum_{i=1}^{\ell} |P_i| + |E|)$.*

Proof. The proof of Lemma 2 in [CCM+21b] has four stages. The technical definitions of these stages are presented in Appendix B of [CCM+21b]. We present the time complexity of the stages.

1. Build the flow reduction $(\mathcal{G}, s, t, d)$ of G (for definition of the flow reduction see Appendix B.3 in [CCM+21b]). The time complexity of this stage is $O(|V| + |E|)$.
2. Build the residual network $\mathcal{R}(\mathcal{G}, \mathcal{P})$ (for definition of the residual network see the proof of Theorem 5 in Appendix B.2 of [CCM+21b]). The time complexity of this stage is $O(\sum_{i=1}^{\ell} |P_i| + |E|)$.
3. Traverse $\mathcal{R}(\mathcal{G}, \mathcal{P})$ and build the set S of all vertices reachable from s. The time complexity of this stage is $O(|V| + |E|)$.
4. Finally, one can compute the maximum antichain of G by inspecting the set S as described in Lemma 2 of [CCM+21b] in additional time of $O(|V| + |E|)$.

■

Completing the Proof of Theorem 1.6. By applying the Algorithm MinimumChainCover on G, we get an MPC $\mathcal{P}$ of $G' = (G \cup H_0)$, such that $\text{TotLen}(\mathcal{P}) = O(n^{3/2})$. Recall that H_0 is the $\sqrt{n}$-shortcut of G. The lemma follows by applying Lemma B.6 on $\mathcal{P}$ and graph G'. ■

4.2 Faster Chain-Antichain Decomposition

For the purpose of this paragraph, let G-chain be a dipath in G and G-antichain be an independent set in G. Recall that an (a, b) decomposition of a DAG G is partitioning of $V(G)$ into a collection of a vertex-disjoint G-chains (i.e., dipaths) and b vertex-disjoint G-antichains (i.e., independent sets), see Def. 1.1. We observe that by reduction to the min-cost max-flow problem one can compute an $(\ell, 2n/\ell)$ decomposition in G in time $\widetilde{O}(n^{3/2} + m)$. We need the following result by [Mir71].

Theorem 4.7. *Suppose that a DAG G has no path of length p, then G has chromatic number at most p. In addition, there is an algorithm, denoted by $\mathsf{Chromatic}(G)$, then one can color $TC(G)$ with at most p colors in time $O(|V(G)| + |E(G)|)$.*

Proof of Theorem 1.7. **The Algorithm.** We consider a (simplified) flow-instance $\mathcal{G} = (\mathcal{V}, \mathcal{E}, u, c)$ where

$$\mathcal{V} = \{v_i^{in}, v_i^{out}, i \in \{1, \ldots, n\}\} \cup \{s, s', t\} .$$

Connect the out-copy v_j^{out} of every incoming neighbor $v_j \in N_{in}(v_i, G)$ to v_i^{in}. In the same manner, connect the in-copy v_k^{in} of every $v_k \in N_{out}(v_i, G)$ to v_i^{out}. Formally, $e_i = (v_i^{in}, v_i^{out})$ and

$$E_1 = \{(v_j^{out}, v_i^{in}) \mid v_j \in N_{in}(v_i, G)\} \text{ and } E_2 = \{(v_j^{in}, v_i^{out}) \mid v_j \in N_{out}(v_i, G)\} .$$

Let $E_3 = \{(s', v_i^{in}) \mid v_i \in V(G)\} \cup \{(v_i^{out}, t) \mid v_i \in V(G)\} \cup \{(s, s'), (s', t)\}$. Then

$$\widetilde{E} = E_1 \cup E_2 \cup E_3 \cup \{e_i | i \in \{1, \ldots, n\}\}.$$

The edge capacities $u \in \mathbb{Z}_{\geq 0}^{\widetilde{E}}$ are defined by $u(s, s') = \ell$ and $u(v_i^{in}, v_i^{out}) = 1$ for every $v_i \in V$. All remaining edges e in $\widetilde{E}$ have capacity of $u(e) = \ell + 1$ (i.e., arbitrarily large). The costs $c \in \mathbb{Z}^{\widetilde{E}}$ are defined by $c(v_i^{in}, v_i^{out}) = -1$, and all remaining edges e in $\widetilde{E}$ have cost of $c(e) = 0$. Let $x = \mathsf{MinCostFlow}(\widetilde{G}, s, t)$, and $\mathcal{P}'$ be the flow-decomposition of x. The output collection of dipaths is given by $\mathcal{P} = \{\mathsf{Translate}(P), P \in \mathcal{P}'\}$. Letting $U = V \setminus V(\mathcal{P})$. Apply Thm. B.7 to color $G[U]$ with $\chi = 2n/\ell$ colors and let $\mathcal{Q} = \{Q_1, \ldots, Q_\chi\}$ the corresponding color classes $G[U]$.

Analysis. Since the capacity of each edge e_i is 1, the paths of $\mathcal{P}$ are vertex-disjoint. In addition, the multiset $\mathcal{P}'$ consists of exactly ℓ paths. We first observe that if $U \neq \emptyset$, then the cost of each path $P' \in \mathcal{P}'$ is *negative*. Assume otherwise that there is $v_i \in U$ and in addition, the 0-cost path $[s, s', t] \in \mathcal{P}'$, then the alternative flow solution in which we send a unit flow over the path $[s, s', v_i^{in}, v_i^{out}, t]$ instead of sending over $[s, s', t]$, provides a legal flow of the same value and strictly lower cost. Consequently, we conclude that if $U \neq \emptyset$, then $[s, s', t] \notin \mathcal{P}'$, and $|\mathcal{P}| = \ell$. That is, $\mathcal{P}$ consists of ℓ vertex-disjoint dipaths.

The key argument is in showing that $\mathtt{Diam}(G[U]) \leq 2n/\ell$, and therefore this graph can be colored with $2n/\ell$ colors. Since $|\mathcal{P}| = \ell$, and as its paths are vertex disjoint, there is a path $\widehat{P} \in \mathcal{P}$ such that $|\widehat{P}| \leq n/\ell$ and thus $c(\mathsf{Translate}^{-1}(P)) \geq n/\ell$. Assume towards contradiction otherwise that $\mathtt{Diam}(G[U]) \geq 2n/\ell + 1$, and let $P' \subseteq G[U]$ be a dipath of length $2n/\ell + 1$. We claim that we can define a path $Q' \subseteq \widetilde{G}$ such that by consider a flow x' obtained by sending one unit flow through Q' and removing a unit flow from $\mathsf{Translate}^{-1}(P)$, one gets a flow of the same value and of strictly lower cost. Let $P' = [v_1, \ldots, v_k]$, then consider the s-t path $Q' = [s, s', v_1^{out}] \circ (v_1^{out}, v_2^{in}) \circ \ldots \circ (v_k^{out}, t)$. Note that $c(Q') \leq -|P'| \leq -(2n/\ell) < c(\mathsf{Translate}^{-1}(P))$.

The alternative flow solution x' is given by $x'(e) = x(e) - 1$ for every $e \in \mathsf{Translate}^{-1}(P)$, $x'(e) = x(e) + 1$ for every $e \in Q'$, and $x'(e) = x(e)$ otherwise. By the definition of the set U, it holds that $x(e_i) = 0$ for every $v_i \in U$, and therefore x' is a legal flow, with the same value as x, but of strictly lower cost, leading to a contradiction. We conclude that $\mathtt{Diam}(G[U]) \leq 2n/\ell$, as desired. Finally, the computation time is given by $\widetilde{O}(m + n^{3/2} + \mathrm{TotLen}(\mathcal{P})) = \widetilde{O}(m + n^{3/2})$, which follows as the $\mathcal{P}$ paths are vertex-disjoint. ■

References

[AMV21] Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. Faster sparse minimum cost flow by electrical flow localization. *CoRR*, abs/2111.10368, 2021.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.

[BGW20] Aaron Bernstein, Maximilian Probst Gutenberg, and Christian Wulff-Nilsen. Near-optimal decremental SSSP in dense weighted digraphs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1112–1122. IEEE, 2020.

[BRR10] Piotr Berman, Sofya Raskhodnikova, and Ge Ruan. Finding sparser directed spanners. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 424–435. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[CC08] Yangjun Chen and Yibin Chen. An efficient algorithm for answering graph reachability queries. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, pages 893–902. IEEE Computer Society, 2008.

[CC14] Yangjun Chen and Yibin Chen. On the graph decomposition. In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing, BDCloud 2014, Sydney, Australia, December 3-5, 2014*, pages 777–784. IEEE Computer Society, 2014.

[CCM+21a] Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I. Tomescu. A linear-time parameterized algorithm for computing the width of a DAG. In Lukasz Kowalik, Michal Pilipczuk, and Pawel Rzazewski, editors, *Graph-Theoretic Concepts in Computer Science - 47th International Workshop, WG 2021, Warsaw, Poland, June 23-25, 2021, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2021.

[CCM+21b] Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I Tomescu. Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. *arXiv preprint arXiv:2107.05717*, 2021.

[Dil50] RP Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, pages 161–166, 1950.

[Fin20] Jeremy T. Fineman. Nearly work-efficient parallel algorithm for digraph reachability. *SIAM J. Comput.*, 49(5), 2020.

[FN18] Sebastian Forster and Danupon Nanongkai. A faster distributed single-source shortest paths algorithm. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 686–697. IEEE Computer Society, 2018.

[GIL+21] Fabrizio Grandoni, Giuseppe F. Italiano, Aleksander Lukasiewicz, Nikos Parotsidis, and Przemyslaw Uznanski. All-pairs LCA in dags: Breaking through the $O(n^{2.5})$ barrier. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 273–289. SIAM, 2021.

[GW20] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Decremental SSSP in weighted digraphs: Faster and against an adaptive adversary. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2542–2561. SIAM, 2020.

[Hes03] William Hesse. Directed graphs requiring large numbers of shortcuts. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 665–669. ACM/SIAM, 2003.

[HKN15] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 725–736. Springer, 2015.

[HP18] Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. In David Eppstein, editor, *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018, June 18-20, 2018, Malmö, Sweden*, volume 101 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Jag90] H. V. Jagadish. A compression technique to materialize transitive closure. *ACM Trans. Database Syst.*, 15(4):558–598, 1990.

[KP22] Shimon Kogan and Merav Parter. New diameter-reducing shortcuts and directed hopsets: Breaking the $\sqrt{n}$ barrier. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2022*, 2022.

[KPG+18] Anna Kuosmanen, Topi Paavilainen, Travis Gagie, Rayan Chikhi, Alexandru I. Tomescu, and Veli Mäkinen. Using minimum path cover to boost dynamic programming on dags: Co-linear chaining extended. In Benjamin J. Raphael, editor, *Research in Computational Molecular Biology - 22nd Annual International Conference, RECOMB 2018, Paris, France, April 21-24, 2018, Proceedings*, volume 10812 of *Lecture Notes in Computer Science*, pages 105–121. Springer, 2018.

[KS97] Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *J. Algorithms*, 25(2):205–220, 1997.

[LJS19] Yang P. Liu, Arun Jambulapati, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1664–1686. IEEE Computer Society, 2019.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433. IEEE Computer Society, 2014.

[Mir71] Leon Mirsky. A dual of dilworth's decomposition theorem. *The American Mathematical Monthly*, 78(8):876–877, 1971.

[MTK+19] Veli Mäkinen, Alexandru I. Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on dags with small width. *ACM Trans. Algorithms*, 15(2):29:1–29:21, 2019.

[Ras10] Sofya Raskhodnikova. Transitive-closure spanners: A survey. In Oded Goldreich, editor, *Property Testing - Current Research and Surveys*, volume 6390 of *Lecture Notes in Computer Science*, pages 167–196. Springer, 2010.

[Tar72] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[Tho92] Mikkel Thorup. On shortcutting digraphs. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science, 18th International Workshop, WG '92, Wiesbaden-Naurod, Germany, June 19-20, 1992, Proceedings*, volume 657 of *Lecture Notes in Computer Science*, pages 205–211. Springer, 1992.

[UY91] Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991.

[vdBGJ+21] Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P. Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. *CoRR*, abs/2112.00722, 2021.

[vdBLL+21] Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 859–869. ACM, 2021.

A Missing Proofs

Proof of Lemma 1.11. Denote the flow value of x by $k = \sum_{u \in N_{out}(s)} x((s,u))$. The algorithm works in k iterations, where in each iteration i, given is a (remaining) valid flow vector x^i where $x^1 = x$. It is assumed that each vertex v stores a pointer $p_i(v)$ to a linked list of its outgoing neighbors u satisfying that $x^i((u,v)) \geq 1$. The output of that iteration is an s-t path P_i and a flow vector x^{i+1} (as well as updated pointers based on x^{i+1}). Let $P_{i,j}$ denote the j'th vertex of path P_i

Iteration $i \geq 1$ (given x^i): Let $x^{i+1} = x^i$ and $j = 1$. Starting with $P_{i,1} = s$, do as follows as long as $P_{i,j} \neq t$. Let $P_{i,j+1}$ be the first outgoing neighbor of $P_{i,j}$ (to which it points), if no such exists, abort and continues to the next iteration. Letting $e_{i,j} = (P_{i,j}, P_{i,j+1})$, then set $x^{i+1}(e_{i,j}) = x^i(e_{i,j}) - 1$. In the case where $x^{i+1}(e_{i,j}) = 0$, omit $P_{i,j+1}$ from the outgoing neighbor list of $P_{i,j}$, and update its pointer accordingly.

Correctness. We show that the algorithm outputs a multi-set of s-t paths $P_1, \ldots, P_k$ each of weight $w_i = 1$ (this can be easily transformed into the desired output of flow decomposition).

We first observe that each output path P_j is an s-t dipath. Since G is a DAG and as the algorithm only proceeds along outgoing edges, it remains to show that P_j ends in t. Assume towards contradiction that it ends at $t' \neq t$. Letting $e' = (u, t')$ be the last edge of P_j, we have that $x^i(e') \geq 1$. Since x^i is a valid flow for G, it holds that there is at least one edge outgoing $e'' = (t', u')$ such that $x^i(e'') \geq 1$ as well, leading to a contradiction that the algorithm aborted in that iteration.

Next, we show that the output vector x^{i+1} is a valid flow in G. To see this note that for each $v \in V(P_i) \setminus \{s, t\}$, it holds that $\sum_{u \in N^{in}(v)} x^{i+1}((u, v)) = \sum_{u \in N^{in}(v)} x^i((u, v)) - 1$ and $\sum_{u \in N^{out}(v)} x^{i+1}((v, u)) = \sum_{u \in N^{in}(v)} x^i((v, u)) - 1$ (as it has one incoming edge and one outgoing edge on P_i). Since x^i is a valid flow, we have that $\sum_{u \in N^{in}(v)} x^{i+1}((u, v)) = \sum_{u \in N^{out}(v)} x^{i+1}((v, u))$. In the same manner, $\sum_{u \in N^{out}(s)} x^{i+1}((s, u)) = \sum_{u \in N^{out}(s)} x^i((s, u)) - 1$ and $\sum_{u \in N^{in}(t)} x^{i+1}((u, t)) = \sum_{u \in N^{in}(t)} x^i((u, t)) - 1$. Therefore, given that x^i is a valid flow, x^{i+1} is a valid flow as well.

We now claim that the multi-set of paths $P_1, \ldots, P_k$ decomposes the flow. Since each x^i is a valid s-t flow, and the value of the flow is reduced in each iteration by exactly one unit, we have that the algorithm terminates within k iterations. As x^{k+1} is a valid flow of value 0, it holds that $x^{k+1}(e) = 0$ and that $x(e) = |\{P_i \mid e \in P_i, i \in \{1, \ldots, k\}\}|$, as desired.

Running Time. The preprocessing takes $O(m + n)$ time. Then each edge is traversed by the algorithm for $x(e)$ many times. This holds as in each iteration i for which $e \in P_i$ the flow value of e is reduced by 1. Since we update the pointers of the vertices to point at their non-zero outgoing neighbors, the algorithm does not traverse edges e once $x^i(e) = 0$. Therefore, the total time is $O(m + n + \sum_e x_e)$. ∎

Proof of Lemma 1.12. The algorithm iterates over the paths of $\mathcal{P}$ in an arbitrary order and maintain a list L of all vertices appearing in the current chain collection. When handling the ℓ'th path $P_\ell \in \mathcal{P}$, the algorithm considers the collection of vertices $V_\ell = V(P_\ell) \setminus L$ and define a chain C_ℓ whose vertices are V_ℓ appearing in the same order as in P_ℓ. The set of vertices V_ℓ is then added to L. It is easy to see that all properties are satisfied and in addition, the running time is linear in the input size, $O(\sum_{P \in \mathcal{P}} |P|)$ as desired. ∎

Proof of Cor. 2.1. Assume that there is a path Q in $TC(G[V \setminus V(\mathcal{P}])$ of more than $2n/\ell$ vertices. as $Q \in TC(G[V \setminus V(\mathcal{P}])$ there is a path P in G which contains all the vertices of Q in particular this path P has more than $2n/\ell$ vertices in $V \setminus V(\mathcal{P})$ which contradicts property (2). ∎

Proof of Observation 2.1. Assume towards contradiction that there is a path $P' \in \widetilde{\mathcal{P}}$ of positive cost, i.e., $c(P') > 0$. Then consider an alternative flow solution x' obtained by replacing the (at least one) unit flow passing through P' with a path $P'' = (s, s', t)$. Formally, define x' by letting:

$$\begin{cases} x'(e) = x(e) - 1, & \text{for every } \ e \in P', \\ x'(e) = x(e) + 1, & \text{for every } e \in P'', \\ x'(e) = x(e), & \text{otherwise .} \end{cases}$$

It is easy to see that x' is a legit flow and of the same value as x. Since $c(P'') < c(P')$, we get that $\sum_e c(e)x(e) > \sum_e c(e)x'(e)$, and end with contradiction that x is a minimum cost max-flow.

In fact, we can prove in this way a stronger statement. For $\ell \leq \omega(G)$, it must hold that all costs are strictly negative. To see this observe that when $U = V \setminus V(\mathcal{P})$ is non-empty, then the

0-cost path (s, s', t) is not in $\widetilde{\mathcal{P}}$ (as otherwise, this path can be replaced with a negative paths $[s, s', v_i^{in}, v_i^{out}, t]$ for some $v_i \in U$). Therefore, a zero-cost path implies that all vertices are covered, hence $\ell \leq \omega(G)$. ■

Proof Sketch for Lemma 3.1. For a fixed path $Q \in \mathcal{Q}$, we explain how to compute $e(v, Q)$ for every $v \in V$ in $O(m)$ time. Traverse the vertices in a *reverse* topological ordering (i.e., the outgoing neighbors of each vertex v appear before v in the ordering). We compute for each v a value $f(v)$ corresponding to the earliest vertex index on Q such that $v \rightsquigarrow Q[i]$ where $Q[i]$ is the i^{th} vertex on Q. Initially, for every $v \in V(Q)$, set $f(v) = j$ such that $Q[j] = v$. The f values are computed by dynamic programming, where for every vertex v let $f(v) = \min_{u \in N_{out}(v)} f(u)$. The correctness and time complexity are immediate. ■

Proof of Theorem 2.2. By Lemmas 2.4 and 2.3, we conclude that $\mathcal{P}$ is indeed an ℓ-cover. We turn to consider the running time. The computation of the flow instance $\widetilde{G}$ can be done in $O(m)$ time. Note that $\widetilde{G}$ is also a DAG and that $|V(\widetilde{G})| = O(n)$ and also $|E(\widetilde{G})| = O(m)$. Thus, by Lemma 1.10 applying Alg. MinCostFlow on $\widetilde{G}$ can be done in $\widetilde{O}(m + n^{3/2})$ time. By Lemma 1.11 decomposing the flow x into the path collection $\widetilde{\mathcal{P}}$ can be done in $O(m + \sum_{P' \in \widetilde{\mathcal{P}}} |V(P')|)$. By Lemma 2.3, we have that $\sum_{P' \in \widetilde{\mathcal{P}}} |V(P')| \leq \min\{\texttt{Diam}(G), \ell\} \cdot n$. Finally, the translation of the paths into $\mathcal{P}$ can be done in linear time in the total path lengths, i.e., $O(\min\{\texttt{Diam}(G), \ell\} \cdot n)$. Overall, the running time is bounded by $\widetilde{O}(m + n^{3/2} + \min\{\texttt{Diam}(G), \ell\} \cdot n)$. ■

Proof of Lemma 3.3. Let P be a shortest u-v dipath in $G \cup H$. Letting $V^+ = V(\mathcal{C})$, we bound the length of P in two steps, by bounding $|V(P) \cap V^+|$ and $|V(P) \cap (V \setminus V^+)|$. We first claim that $|V(P) \cap (V \setminus V^+)| \leq 2\sqrt{n}$. Note that $V^+ = V(\mathcal{P})$, and therefore the claim follows by property (2) of Definition 2.1. Next, we bound $|V(P) \cap V^+|$, namely, the number of P's vertices that belong to the chains of $\mathcal{C}$.

The key observation is that since $H(C_i) \subseteq H$ for every $C_i \in \mathcal{C}$, and since G is a DAG, it follows that $|V(P) \cap V(C_i)| \leq 3$. This holds since for each $C_i \in \mathcal{C}$, the vertices appearing on P must appear in the same order as in C_i, as G is a DAG, and since there is 2-hop dipath in $H(C_i)$ between every two vertices in C_i. Altogether, we have that $|V(P) \cap V^+| \leq 3 \cdot |\mathcal{C}|$. By Definition 2.1, $|\mathcal{P}| \leq \sqrt{n}$, as $|\mathcal{C}| = |\mathcal{P}|$ we have $|V(P) \cap V^+| \leq 3\sqrt{n}$. The diameter bound follows. ■

Proof of Lemma 3.2. We start with the running time analysis. The computation of the ℓ-cover by Theorem 2.2 for $\ell = \Theta(\sqrt{n})$, takes $\widetilde{O}(m + n^{3/2})$ time. By property (1) of ℓ-covers, and using Lemma 1.12, the computation of the chain collection $\mathcal{C}$ can be done in time $O(\ell \cdot n) = O(n^{3/2})$. Since the chains of $\mathcal{C}$ are vertex-disjoint, we have that $\sum_{C_i \in \mathcal{C}} |C_i| = O(n)$. Therefore, using Lemma 1.8, the computation of the shortcut graphs H_i for each chain $C_i \in \mathcal{C}$ can be done in total time $O(n \log n)$.

We next bound the number of shortcut edges. Since the chains in $\mathcal{C}$ are vertex-disjoint, we have that $|\bigcup_{C_i \in \mathcal{C}} H(C_i)| = O(n \log n)$. ■

Proof of Lemma 3.4. **Running time:** Computing the shortcut set H_0 is done in time $\widetilde{O}(m + n^{3/2})$ by Theorem 1.3. Since $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$, applying Alg. PathCover takes $\widetilde{O}(m + n^{3/2})$ time as well, using Theorem 2.2. By property (2) of Def. 2.1, we have that $\text{TotLen}(\mathcal{P}) = O(\texttt{Diam}(G \cup$

$H_0) \cdot n) = O(n^{3/2})$. Therefore, the computation of $\mathcal{C}$ can be done in time $O(n^{3/2})$ using Lemma 1.12.

Since all chains in $\mathcal{C}$ are vertex-disjoint, we have that the computation of $\bigcup H(C_i)$ can be done in time $O(n \log n)$ using Lemma 1.8. Computing the sampled sets $\mathcal{C}'$ and V' can be done in linear time. Finally, the computation of the set $\widehat{H}$ takes $O(|\mathcal{C}'| \cdot m)$ by using Lemma 3.1. Since $|\mathcal{C}| = |\mathcal{P}| = \ell$, we have that w.h.p. $|\mathcal{C}'| = \widetilde{O}(n/D^2)$. Therefore, this last step takes $\widetilde{O}(m \cdot n/D^2)$. Overall the running time is bounded, w.h.p., by $\widetilde{O}(m \cdot n/D^2 + n^{3/2})$ as required.

Size bound: By Theorem 1.3, $|H_0| = \widetilde{O}(n)$. Since the chains in $\mathcal{C}$ are vertex-disjoint, by Lemma 1.8 we have that $|\bigcup H(C_i)| = O(n \log n)$. Finally, w.h.p. $|V'| = \widetilde{O}(n/D)$ and by the above argument, $|\mathcal{C}'| = \widetilde{O}(n/D^2)$. Therefore, w.h.p., $|\widehat{H}| = \widetilde{O}(n^2/D^3)$, as required. ■

Proof of Lemma 3.5. Let $P'' = [u_1, \ldots, u_\ell = v]$. We consider a corresponding path $Q \in G \cup H_0$ defined based on P'' as follows. Every edge $(u_j, u_{j+1}) \in P''$ that belongs to $\bigcup(C_i \cup H(C_i))$ is replaced in Q by a u-v shortest path in $G \cup H_0$. Note that $V(P'') \subseteq V(Q)$, and since $G \cup H_0$ is a DAG, it holds that Q is a dipath. By property (2) of Definition 2.1 for $\ell = 16n/D$, we have that $|V(Q) \cap (V \setminus V(\mathcal{P}))| \leq D/8$. Since $V(\mathcal{C}) = V(\mathcal{P})$ and $V(P'') \subseteq V(Q)$, we have that $|V(P'') \cap (V \setminus V(\mathcal{C}))| \leq D/8$. ■

A.1 Proof of Theorem 2.6

In this section, we analyze Algorithm PartialPathCover. First observe that by the same argument as in Obs. 2.1, we have that:

Observation A.1. $c(P) < 0$ for any $P \in \widetilde{\mathcal{P}}$.

A vertex v_i in a path $P_j \in \mathcal{P}$ is denoted as a *principal vertex* of P if the corresponding flow path $\mathsf{Translate}^{-1}(P_j) \in \widetilde{\mathcal{P}}$ contains the edge (v_i^{in}, v_i^{out}). Recall that $n(P_j)$ is the number of principal vertices in P_j. Since for every vertex $v_i \in V \setminus V'$, it holds that $u'((v_i^{in}, v_i^{out})) = 0$ (zero capacity edge), it holds that v_i cannot be a principal vertex of any of the paths in $\mathcal{P}$. In addition, since for every $v_i \in V'$ it holds that $u'((v_i^{in}, v_i^{out})) = 1$, we have that each $v_i \in V'$ can be a principal vertex of at most *one* path. Consequently, we have the following:

Observation A.2. $\sum_{P_j \in \mathcal{P}} n(P_j) \leq |V(\mathcal{P}) \cap V'|$.

We are now ready to prove the two main properties of partial ℓ-covers.

Lemma A.1. *$|\mathcal{P}| = \ell$ and $\sum_{P \in \mathcal{P}} |V(P)| \leq 8|V(\mathcal{P}) \cap V'|$.*

Proof. The bound of the number of paths simply follows by the fact that x is an integral flow of value ℓ, and by the definition of flow decomposition (Def. 1.3). We start by showing that $|V(P_j)| \leq 8n(P_j)$ for any path $P_j \in \mathcal{P}$. Assume towards contradiction otherwise. Let $P_j \in \mathcal{P}$ be such that $|V(P_j)| \geq 8n(P) + 1$, and let $P_j' = \mathsf{Translate}^{-1}(P_j)$ be the corresponding path in the flow-decomposition $\widetilde{\mathcal{P}}$.

Since $|V(P_j)| \geq 8n(P_j) + 1$, we get that P_j' contains at least $7n(P_j) + 1$ edges of positive costs 1, and at most $n(P_j)$ edges of negative cost (one per principal vertex). Therefore, $c(P_j') \geq 7n(P_j) +$

$1 - 7n(P_j) > 0$, leading to a contradiction by Observation A.1. By combining with Obs. A.2, we get that:

$$\sum_{P_j \in \mathcal{P}} |V(P_j)| \leq \sum_{P_j \in \mathcal{P}} 8n(P_j) \leq 8|V(\mathcal{P}) \cap V'| \,.$$

■

Lemma A.2. *For any dipath $Q \subseteq G$ such that $|Q| \geq 8|V' \cap V(\mathcal{P})|/\ell$, it holds that $|V(Q) \cap (V' \setminus V(\mathcal{P}))| \leq |Q|/4$.*

Proof. Observe that as $|\mathcal{P}| = \ell$ and by Obs. A.2, there exists a path $P^* \in \mathcal{P}$ with $n(P^*) \leq |V' \cap V(\mathcal{P})|/\ell$ principal vertices. Therefore,

$$c(\mathsf{Translate}^{-1}(P^*)) \geq -7|V' \cap V(\mathcal{P})|/\ell \,. \tag{6}$$

Let $U = V' \setminus V(\mathcal{P})$ be the subset of uncovered vertices in V' by the paths of $\mathcal{P}$. Assume towards contradiction that there exists a path $Q \subseteq G$ of length at at least $r \geq 8|V' \cap V(\mathcal{P})|/\ell$ such that $|V(Q) \cap U| > r/4$. We next use $Q = [v_1, \ldots, v_r]$ (a dipath in G) to define a dipath $Q' \subseteq \widetilde{G}$:

$$Q' = T_1 \circ (v_1^{out}, v_2^{in}) \circ T_2 \circ (v_2^{out}, v_3^{in}) \circ T_3 \ldots \circ T_r \,,$$

where $T_j = (v_j^{in}, v_j^{out})$ if $v_j \in U$, and $T_j = [v_j^{in}, v_j', v_j^{out}]$ otherwise. We then have that

$$c(Q') \leq -7 \cdot (r/4) + 1 \cdot (3r/4) \leq -r \leq -8|V' \cap V(\mathcal{P})|/\ell \,. \tag{7}$$

Consider an alternative flow solution x' obtained by sending one unit flow on Q' rather than on $\mathsf{Translate}^{-1}(P^*)$. Formally, define x' by letting:

$$\begin{cases} x'(e) = x(e) - 1, & \text{for } e \in \mathsf{Translate}^{-1}(P^*), \\ x'(e) = x(e) + 1, & \text{for } e \in Q', \\ x'(e) = x(e), & \text{otherwise} \,. \end{cases}$$

We claim that x' is a legal flow of strictly lower cost. By the definition of U, $x((v_i^{in}, v_i^{out})) = 0$ for every $v_i \in U$, and therefore, we indeed have the capacity to send one unit flow along Q'. In addition, by Eq. (6, 7), $c(\mathsf{Translate}^{-1}(P^*)) > c(Q')$, concluding that $\sum_e c(e)x(e) > \sum_e c(e)x'(e)$, leading to a contradiction, as desired. ■

We are now ready to complete the proof of Theorem 2.6.

Proof of Theorem 2.6. By Lemma A.1 and A.2, it follows that the $\mathcal{P}$ is indeed a partial ℓ-cover with respect to V'. It remains to bound the running time. Computing the min-cost-max-flow takes $\widetilde{O}(m + n^{3/2})$ by Theorem 1.10. By Lemma 1.11, computing the flow-decomposition $\widetilde{\mathcal{P}}$ takes $O(\sum_{P \in \widetilde{\mathcal{P}}} |V(P)|)$ time. Since $|V(P)| = O(|V(\mathsf{Translate}(P))|$ for every $P \in \widetilde{\mathcal{P}}$, by combining with property (2) of Def. 2.2, we have that this is done in time $O(|V'|)$. The last step of computing $\mathsf{Translate}(P)$ for every $P \in \widetilde{\mathcal{P}}$ takes $O(|V'|)$ time as well. ■

B Applications

B.1 Minimum Chain Covers and Maximum Antichains

Recall that a chain is dipath in $TC(G)$ and an antichain is a subset of vertices that are independent set in $TC(G)$. In this section, we prove Theorem 1.6. For a dipath collection $\mathcal{P}$, let $\text{TotLen}(\mathcal{P}) = \sum_{P\in\mathcal{P}} |P|$. Note that it might be the case that $\text{TotLen}(\mathcal{P}) \gg |V(\mathcal{P})|$ (in case where the dipaths are not vertex-disjoint, and each vertex appears on multiple paths).

We need the following definitions.

Definition B.1 (Minimum Path Covers)**.** A minimum path cover (MPC) of a DAG $G = (V, E)$ is a minimum-size set of dipaths in G that covers V.

Lemma B.1. *Given an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ for a DAG $G = (V, E)$, the chain collection $\mathcal{C} = \mathsf{DisjointChains}(\mathcal{P})$ is an MCC. Given MPC, the MCC can be thus computed in time $O(\text{TotLen}(\mathcal{P}))$.*

The MCC algorithm is based on combining the following three ingredients:

1. Computing an MPC $\mathcal{P}$ (consisting of G-dipaths) in time $\widetilde{O}(m + n^{3/2} + \text{TotLen}(\mathcal{P}))$.
2. Showing that every digraph G has an MPC $\mathcal{P}$ of $\text{TotLen}(\mathcal{P}) \leq n \cdot \texttt{Diam}(G)$.
3. Computing a *Minimum-Length* MPC $\mathcal{P}$ in time $\widetilde{O}(m + n^{3/2} + \text{TotLen}(\mathcal{P}))$.

In Sec. B.1.1 we introduce the key tool of computing MPC of minimum length, and then in Sec. B.1.2 we provide the detailed MCC algorithm.

B.1.1 Minimum Length Cover

We start by introducing the notion of minimum-length covers and then show how to compute them as a function of their length.

Definition B.2 (Minimum Length Covers)**.** For a DAG G and input parameter $\ell \geq \omega(G)$, an ℓ-Minimum Length Cover (MLC) is a G-dipath collection $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ satisfying that $V(\mathcal{P}) = V$ (i.e., $\mathcal{P}$ is a path-cover). In addition, $\text{TotLen}(\mathcal{P})$ is the minimum over any other path cover of cardinality ℓ. That is, for any other collection of G-dipaths $\mathcal{P}'$ that covers V and $|\mathcal{P}'| = \ell$, it holds that $\text{TotLen}(\mathcal{P}') \geq \text{TotLen}(\mathcal{P})$.

We next present an algorithm that given a DAG G and a parameter ℓ, either reports that $\omega(G) > \ell$, or else outputs an ℓ-MLC. Algorithm MinimumLengthCover starts by computing the min-cost max-flow $x \in \mathbb{R}^{|\widetilde{E}|}$ of the graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$ (defined as in Alg. PathCover). The case where $\ell < \omega(G)$ is detected by the algorithm based on the cost of the flow. Specifically, in the case where $c(x) = \sum_{e\in\widetilde{E}} c(e)x(e) \geq n^3 - n^4$, the algorithm determines that $\ell < \omega(G)$ and aborts. Intuitively, since the edges (v_i^{out}, v_i^{in}) have negative cost of $-n^3$, in case where the flow solution x *covers* all vertices, it should have a sufficiently small cost. Alternatively, in the case where $\ell \geq \omega(G)$, the algorithm proceeds as follows. It computes the flow-decomposition of x, which we denote by $\mathcal{P}$. The output ℓ-MLC is given by the translated G-paths that correspond to $\mathcal{P}$. This completes the description of the algorithm.

Algorithm MinimumLengthCover

Input: An n-vertex DAG G and $\ell \in \mathbb{N}_{\geq 1}$.
Output: For $\ell < \omega(G)$, return -1. Otherwise, return an ℓ-MLC.

1. Let $\widetilde{G} = (\widetilde{V}, \widetilde{E}, u, c)$ be defined as in Alg. PathCover (Theorem 2.2).
2. Apply Algorithm MinCostFlow$(\widetilde{G}, s, t)$ (Theorem 1.10), and let $x \in \mathbb{R}^{|\widetilde{E}|}$ be the output flow.
3. If $c(x) \geq n^3 - n^4$, return -1.
4. **Flow Decomposition.** Decompose x into a multiset of s-t paths $\mathcal{P}'$ in $\widetilde{G}$ by applying Lemma 1.11.
5. Return $\mathcal{P} = \{\mathsf{Translate}(P') \mid P' \in \mathcal{P}'\}$.

Correctness. The minimality of the cover follows by Theorem 2.5. Let ℓ be the input parameter to MinimumLengthCover, and recall $c(x) = \sum_{e \in \widetilde{E}} x(e) \cdot c(e)$ be the cost of the output flow $x = \mathsf{MinCostFlow}(\widetilde{G}, s, t)$.

Observation B.1. If $\omega(G) \leq \ell$, then $c(x) \leq n^2 - n^4$, and otherwise $c(x) \geq n^3 - n^4$.

Proof. Assume that $\omega(G) \leq \ell$. Observe that in this case $x(v_i^{in}, v_i^{out}) = 1$ for every $i \in \{1, \ldots, n\}$. Since the value of the flow is ℓ and $c(P') \leq n$ for every dipath $P' \subseteq \widetilde{G}$, we have that $c(x) \leq \sum_{i=1}^{n} c(v_i^{in}, v_i^{out}) \cdot x(v_i^{in}, v_i^{out}) + n^2 \leq n^2 - n^4$.

In the case where $\omega(G) > \ell$, there is at least one vertex v_i that is uncovered by the paths $\mathcal{P}$. This implies that $x(v_i^{in}, v_i^{out}) = 0$, and therefore we have that $c(x) \geq -(n-1)n^3$. ■

Claim B.2. *If $\ell \geq \omega(G)$, then* $\mathrm{TotLen}(\mathcal{P}) = c(x) + n^4 + n$.

Proof. This follows from the fact that there are n edges of the form (v_i^{in}, v_i^{out}) and $\sum_{i=1}^{n} c(v_i^{in}, v_i^{out}) \cdot x(v_i^{in}, v_i^{out}) = -n^4$ and the rest of the edges that contribute to M are of the form (v_i^{in}, v_i') for which $c(v_i^{in}, v_i') = 1$. That is, we have the following identity:

$$\sum_{i=1}^{n} c(v_i^{in}, v_i') \cdot x(v_i^{in}, v_i') + n = \sum_{i=1}^{n} x(v_i^{in}, v_i') + n = \sum_{i=1}^{k} |P_i| \,.$$

The claim follows by the optimality of MinCostFlow (i.e., that x is of minimum-cost). ■

The running time is $\tilde{O}(m + n^{3/2} + \mathrm{TotLen}(\mathcal{P}))$ by Theorem 1.10.

B.1.2 Algorithm for Computing Minimum Chain Covers

We are now ready to describe the MCC algorithm that establishes Theorem 1.6. The algorithm starts by reducing the diameter of G to $\sqrt{n}$ by computing a $\sqrt{n}$-shortcut set H_0 (using Theorem 1.3). It then computes an $\omega(G)$-MLC $\mathcal{P}$. Note that the dipaths of $\mathcal{P}$ are in $G \cup H_0$ and are not

necessarily vertex-disjoint. Finally, the algorithm transforms the $\omega(G)$-MLC into an MCC using Alg. DisjointChains. Since the complexity of the latter algorithm depends on $\text{TotLen}(\mathcal{P})$, the runtime argument will (i) exploit the fact that $\mathcal{P}$ has minimum total-length, and (ii) provide an upper on the total length of an ℓ-MLC.

Algorithm MinimumChainCover**:**

Input: An n-vertex DAG G with m edges.
Output: An MCC of G.

1. Let $H_0 \leftarrow \text{FasterShortcutSqrtN}(G)$ where $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$ (using Theorem 1.3), and set $G' = G \cup H_0$.
2. Compute $\omega(G')$ by Binary search by applying Alg. $\text{MinimumLengthCover}(G', \ell)$ (where ℓ is the width guess).
3. $\mathcal{P} = \text{MinimumLengthCover}(G', \omega(G'))$.
4. Output $\mathcal{C} = \text{DisjointChains}(\mathcal{P})$

Analysis. Observe that $\omega(G) = \omega(G')$ as $H_0 \subseteq TC(G)$. Since an $\omega(G)$-Minimum-Length-Cover $\mathcal{P}$ is a path cover with cardinality $\omega(G)$, we have that:

Claim B.3 (Correctness). *$\mathcal{C}$ is an MCC, w.h.p.*

To bound the running time, we need the following lemma.

Lemma B.4. *Any DAG G has an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of total length* $\text{TotLen}(\mathcal{P}) \leq n \cdot \texttt{Diam}(G)$.

Proof. Let $\mathcal{C}$ be some MCC of G (see Lemma B.1). Since the chains in $\mathcal{C}$ are vertex-disjoint, we have that $\sum_{C_i \in \mathcal{C}} |C_i| \leq n$. We transform each chain C_i into a path $P_i \subseteq G$. This can be done by traversing C_i and replacing each $(u, v) \in C_i \setminus G$ by the u-v shortest path $P_{u,v}$ in G. Formally, letting $C_i = [u_1, \ldots, u_q]$ then $P_i = P_{u_1,u_2} \circ P_{u_2,u_3} \circ \ldots \circ P_{u_{q-1},u_q}$. Since G is a DAG, P_i is a dipath in G. Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ where $k = \omega(G)$.

As all chains in $\mathcal{C}$ are vertex-disjoint and covering V, we have that $\mathcal{P}$ is an MPC (as each path P_i is contained in G). In addition, since each edge (u, v) in C_i is replaced in P_i by a u-v shortest path in G, we have that $|P_i| \leq |C_i| \cdot \texttt{Diam}(G)$, as . Therefore,

$$\sum_{i=1}^{k} |P_i| \leq \texttt{Diam}(G) \cdot \sum_{i=1}^{k} |C_i| \leq \texttt{Diam}(G) \cdot n.$$

■

Claim B.5. *Algorithm* MinimumChainCover *can be implemented in time $\widetilde{O}(m + n^{3/2})$.*

Proof. The computation of the shortcut H_0 takes $\widetilde{O}(m + n^{3/2})$ time by Theorem 1.3. Each application of Alg. $\text{MinimumLengthCover}(G', \ell)$ takes $\widetilde{O}(m + n^{3/2})$ (by Lemma B.4 and the fact that $\texttt{Diam}(G \cup H_0) = O(\sqrt{n})$) and there are $O(\log n)$ such applications. By the correction of Alg. MinimumLengthCover, once again we have by Lemma B.4 that $\sum_{i=1}^{k} |P_i| \leq \texttt{Diam}(G \cup H_0) \cdot n = O(n^{3/2})$. Therefore, the computation of $\mathcal{C}$ takes $O(n^{3/2})$ time. ■

B.1.3 Maximum Antichains

Recall that an antichain of a DAG G is a set of pairwise non-reachable vertices in G. Our goal is to compute an antichain of maximum size. The next lemma follows immediately from the proof of Lemma 2 in [CCM+21b].

Lemma B.6. *[Restatement of Lemma 2 in [CCM+21b]] Given a DAG $G = (V, E)$ and an MPC $\mathcal{P} = \{P_1, P_2, \ldots, P_\ell\}$, one can compute a maximum antichain of G in time $O(\sum_{i=1}^{\ell} |P_i| + |E|)$.*

Proof. The proof of Lemma 2 in [CCM+21b] has four stages. The technical definitions of these stages are presented in Appendix B of [CCM+21b]. We present the time complexity of the stages.

1. Build the flow reduction $(\mathcal{G}, s, t, d)$ of G (for definition of the flow reduction see Appendix B.3 in [CCM+21b]). The time complexity of this stage is $O(|V| + |E|)$.
2. Build the residual network $\mathcal{R}(\mathcal{G}, \mathcal{P})$ (for definition of the residual network see the proof of Theorem 5 in Appendix B.2 of [CCM+21b]). The time complexity of this stage is $O(\sum_{i=1}^{\ell} |P_i| + |E|)$.
3. Traverse $\mathcal{R}(\mathcal{G}, \mathcal{P})$ and build the set S of all vertices reachable from s. The time complexity of this stage is $O(|V| + |E|)$.
4. Finally, one can compute the maximum antichain of G by inspecting the set S as described in Lemma 2 of [CCM+21b] in additional time of $O(|V| + |E|)$.

■

Completing the Proof of Theorem 1.6. By applying the Algorithm MinimumChainCover on G, we get an MPC $\mathcal{P}$ of $G' = (G \cup H_0)$, such that $\text{TotLen}(\mathcal{P}) = O(n^{3/2})$. Recall that H_0 is the $\sqrt{n}$-shortcut of G. The lemma follows by applying Lemma B.6 on $\mathcal{P}$ and graph G'. ■

B.2 Faster Chain-Antichain Decomposition

For the purpose of this paragraph, let G-chain be a dipath in G and G-antichain be an independent set in G. Recall that an (a, b) decomposition of a DAG G is partitioning of $V(G)$ into a collection of a vertex-disjoint G-chains (i.e., dipaths) and b vertex-disjoint G-antichains (i.e., independent sets), see Def. 1.1. We observe that by reduction to the min-cost max-flow problem one can compute an $(\ell, 2n/\ell)$ decomposition in G in time $\widetilde{O}(n^{3/2} + m)$. We need the following result by [Mir71].

Theorem B.7. *Suppose that a DAG G has no path of length p, then G has chromatic number at most p. In addition, there is an algorithm, denoted by $\mathsf{Chromatic}(G)$, then one can color $TC(G)$ with at most p colors in time $O(|V(G)| + |E(G)|)$.*

Proof of Theorem 1.7. **The Algorithm.** We consider a (simplified) flow-instance $\mathcal{G} = (\mathcal{V}, \mathcal{E}, u, c)$ where

$$\mathcal{V} = \{v_i^{in}, v_i^{out}, i \in \{1, \ldots, n\}\} \cup \{s, s', t\} .$$

Connect the out-copy v_j^{out} of every incoming neighbor $v_j \in N_{in}(v_i, G)$ to v_i^{in}. In the same manner, connect the in-copy v_k^{in} of every $v_k \in N_{out}(v_i, G)$ to v_i^{out}. Formally, $e_i = (v_i^{in}, v_i^{out})$ and

$$E_1 = \{(v_j^{out}, v_i^{in}) \mid v_j \in N_{in}(v_i, G)\} \text{ and } E_2 = \{(v_j^{in}, v_i^{out}) \mid v_j \in N_{out}(v_i, G)\} .$$

Let $E_3 = \{(s', v_i^{in}) \mid v_i \in V(G)\} \cup \{(v_i^{out}, t) \mid v_i \in V(G)\} \cup \{(s, s'), (s', t)\}$. Then

$$\widetilde{E} = E_1 \cup E_2 \cup E_3 \cup \{e_i | i \in \{1, \ldots, n\}\}.$$

The edge capacities $u \in \mathbb{Z}_{\geq 0}^{\widetilde{E}}$ are defined by $u(s, s') = \ell$ and $u(v_i^{in}, v_i^{out}) = 1$ for every $v_i \in V$. All remaining edges e in $\widetilde{E}$ have capacity of $u(e) = \ell + 1$ (i.e., arbitrarily large). The costs $c \in \mathbb{Z}^{\widetilde{E}}$ are defined by $c(v_i^{in}, v_i^{out}) = -1$, and all remaining edges e in $\widetilde{E}$ have cost of $c(e) = 0$. Let $x = \mathsf{MinCostFlow}(\widetilde{G}, s, t)$, and $\mathcal{P}'$ be the flow-decomposition of x. The output collection of dipaths is given by $\mathcal{P} = \{\mathsf{Translate}(P), P \in \mathcal{P}'\}$. Letting $U = V \setminus V(\mathcal{P})$. Apply Thm. B.7 to color $G[U]$ with $\chi = 2n/\ell$ colors and let $\mathcal{Q} = \{Q_1, \ldots, Q_\chi\}$ the corresponding color classes $G[U]$.

Analysis. Since the capacity of each edge e_i is 1, the paths of $\mathcal{P}$ are vertex-disjoint. In addition, the multiset $\mathcal{P}'$ consists of exactly ℓ paths. We first observe that if $U \neq \emptyset$, then the cost of each path $P' \in \mathcal{P}'$ is *negative*. Assume otherwise that there is $v_i \in U$ and in addition, the 0-cost path $[s, s', t] \in \mathcal{P}'$, then the alternative flow solution in which we send a unit flow over the path $[s, s', v_i^{in}, v_i^{out}, t]$ instead of sending over $[s, s', t]$, provides a legal flow of the same value and strictly lower cost. Consequently, we conclude that if $U \neq \emptyset$, then $[s, s', t] \notin \mathcal{P}'$, and $|\mathcal{P}| = \ell$. That is, $\mathcal{P}$ consists of ℓ vertex-disjoint dipaths.

The key argument is in showing that $\mathtt{Diam}(G[U]) \leq 2n/\ell$, and therefore this graph can be colored with $2n/\ell$ colors. Since $|\mathcal{P}| = \ell$, and as its paths are vertex disjoint, there is a path $\widehat{P} \in \mathcal{P}$ such that $|\widehat{P}| \leq n/\ell$ and thus $c(\mathsf{Translate}^{-1}(P)) \geq n/\ell$. Assume towards contradiction otherwise that $\mathtt{Diam}(G[U]) \geq 2n/\ell + 1$, and let $P' \subseteq G[U]$ be a dipath of length $2n/\ell + 1$. We claim that we can define a path $Q' \subseteq \widetilde{G}$ such that by consider a flow x' obtained by sending one unit flow through Q' and removing a unit flow from $\mathsf{Translate}^{-1}(P)$, one gets a flow of the same value and of strictly lower cost. Let $P' = [v_1, \ldots, v_k]$, then consider the s-t path $Q' = [s, s', v_1^{out}] \circ (v_1^{out}, v_2^{in}) \circ \ldots \circ (v_k^{out}, t)$. Note that $c(Q') \leq -|P'| \leq -(2n/\ell) < c(\mathsf{Translate}^{-1}(P))$.

The alternative flow solution x' is given by $x'(e) = x(e) - 1$ for every $e \in \mathsf{Translate}^{-1}(P)$, $x'(e) = x(e) + 1$ for every $e \in Q'$, and $x'(e) = x(e)$ otherwise. By the definition of the set U, it holds that $x(e_i) = 0$ for every $v_i \in U$, and therefore x' is a legal flow, with the same value as x, but of strictly lower cost, leading to a contradiction. We conclude that $\mathtt{Diam}(G[U]) \leq 2n/\ell$, as desired. Finally, the computation time is given by $\widetilde{O}(m + n^{3/2} + \mathrm{TotLen}(\mathcal{P})) = \widetilde{O}(m + n^{3/2})$, which follows as the $\mathcal{P}$ paths are vertex-disjoint. ∎