Maximum Coverage k-Antichains and Chains: A Greedy Approach

Manuel Cáceres
Department of Computer Science, Aalto University, Espoo, Finland
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Andreas Grigorjew
LAMSADE, CNRS UMR7243, Université Paris Dauphine-PSL, 75775 Paris, France
Institute of Informatics, University of Warsaw, Poland

Wanchote Po Jiamjitrak
Department of Computer Science, University of Helsinki, Finland

Alexandru I. Tomescu
Department of Computer Science, University of Helsinki, Finland

Abstract

Given an acyclic digraph $G = (V, E)$ and a positive integer k, the problem of Maximum Coverage k-Antichains (resp. Chains) denoted as MA-k (resp. MC-k) asks to find k sets of pairwise unreachable vertices, known as antichains (resp. k subsequences of paths, known as chains), maximizing the number α_k (resp. β_k) of vertices covered by these antichains (resp. chains). While MC-k was solved in almost optimal $|E|^{1+o(1)}$ time [Kogan and Parter, ICALP'22], the fastest algorithms for MA-k are a $(k|E|)^{1+o(1)}$-time solution and a $|E|^{1+o(1)}$-time $1/2$ approximation [Kogan and Parter, ESA'24].

We simplify and improve previous results. Specifically, we obtain the following for MA-k:

- An algorithm running in $|E|^{1+o(1)}$ time, and an algorithm running in *parameterized near-linear* $\tilde{O}(\alpha_k|E|)$ time. Our algorithms are simple solutions exploiting a paths-based proof of the Greene-Kleitman theorems leveraged by the `greedy` algorithm for set cover as well as recent advances in fast algorithms for flows and shortest paths.
- An approximation algorithm running in *parameterized linear* time $O(\alpha_1^2|V| + (\alpha_1 + k)|E|)$ with approximation ratio of $(1 - 1/e) > 0.63 > 1/2$, beating the state-of-the-art $1/2$ approximation. Our solution uses `greedy` for antichains and a simple strategy to amortize the cost of computing consecutive maximum antichains.

Additionally, we obtain analogous results for MC-k as well as the corresponding dual problems derived from the Greene-Kleitman theorems, which might be of independent interest.

We complement these results with two examples (one for chains and one for antichains) showing that, for every $k \geq 2$, `greedy` misses the tight $1/e$ portion of the optimal coverage for chains, and a $1/4$ portion for antichains. We also show that `greedy` is a $\Omega(\log |V|)$ factor away from minimality when required to cover all vertices: previously unknown for sets of chains or antichains.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Network flows; Theory of computation → Sorting and searching

Keywords and phrases Maximum coverage antichains, maximum coverage chains, directed acyclic graph, minimum cost flow, greedy set cover, parameterized algorithms, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.108

Funding Co-funded by the European Union (ERC, SCALEBIO, 101169716). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Co-funded also by the Research Council of Finland grants No. 346968, 358744.
Manuel Cáceres: Supported by the Helsinki Institute for Information Technology HIIT.
Andreas Grigorjew: Supported by ANR project ANR-21-CE48-0022 (S-EX-AP-PE-AL).

Acknowledgements We appreciate the insightful comments provided by the anonymous reviewers.

1 Introduction

For a directed acyclic graph (DAG) $G = (V, E)$, a chain is a subsequence of the vertices of a path (equivalently a path in the transitive closure $TC(G)$) and an antichain is a set of vertices that are pairwise unreachable (equivalently, an independent set in $TC(G)$). The problem of Maximum Coverage k-antichains (MA-k) asks to find a set of k pairwise disjoint antichains whose union size is maximized: we say that one such optimal solution *covers* the α_k vertices in this union. The problem of Maximum Coverage k-chains (MC-k) is analogously defined and every optimal solution covers β_k vertices.

Strongly related to these coverage problems are the minimum partition problems: find the minimum number of pairwise disjoint antichains (analogously chains) partitioning the vertices of G, MAP (resp. MCP). Mirsky [31] and Dilworth [13] were the first to draw a connection between the maximum coverage and minimum partition problems. On the one hand, Mirsky proved that the number of antichains in a MAP equals β_1, also known as the *height* of G [31]. On the other hand, Dilworth proved that the number of chains in a MCP equals α_1, also known as the *width* of G [13]. Later, Greene and Kleitman [22, 21] completed this connection with the celebrated Greene-Kleitman (GK) theorems[1]: α_k (resp. β_k) is equal to the minimum k-*norm* of a chain (resp. antichain) partition[2]. We denote partitions with minimum k-norm by MCP-k and MAP-k, respectively.

From an algorithmic perspective, on the one hand, it is a well-known result that we can solve MAP and MC-1 in linear time: by a simple DP one can compute the length of the longest path ending at each vertex (solving MC-1), moreover a partition of the vertices according to these lengths gives a MAP. On the other hand, Fulkerson [18] was the first to show a polynomial-time algorithm for MCP and MA-1 by reducing the problem to maximum bipartite matching. However, Fulkerson's algorithm computes a maximum matching over $TC(G)$. Modern solutions avoid working directly on $TC(G)$ since its size can be quadratic in G. The state-of-the-art solutions for MCP and MA-1 are the almost linear $|E|^{1+o(1)}$-time solution of Cáceres [5], the parameterized linear $O(\alpha_1^2|V| + |E|)$-time algorithm of Cáceres et al. [9, 8] and the parameterized near-linear $\tilde{O}(\alpha_1|E|)$-time of Mäkinen et al. [30]. The latter computes a `greedy` chain partition, which is then shrunk to minimum size. In fact, this `greedy` solution is exactly the output of the well-known `greedy` set cover [12] with sets corresponding to the chains of the DAG and thus it inherits the $O(\log n)$ approximation ratio. This idea was originally proposed by Felsner et al. [15] for transitive DAGs, and it was later optimized for general DAGs [28, 30], avoiding the computation of $TC(G)$.

For larger k, MC-k [26, 5] has been solved in almost optimal $|E|^{1+o(1)}$ time via a reduction to Minimum Cost Flow. However, until recently, the fastest solution for MA-k was an $O(|V|^3)$-time algorithm from the 80's by Gavril [19]. The big difference in the running times between both problems was tantalizing and Kogan and Parter [27] tackled this gap by exploiting algorithmic properties of the GK theorems. They present an elegant 1/2 approximation running in time $|E|^{1+o(1)}$, as well as an approximation scheme (only for transitive DAGs) running in time $O(\alpha_k\sqrt{|E|}|V|^{o(1)}/\epsilon + |V|)$. The same paper also shows an exact solution running in $(k|E|)^{1+o(1)}$ time beating the $O(|V|^3)$-time algorithm of Gavril.

The original proof of the GK theorems by Greene and Kleitman [22, 21] was based on lattice theoretic methods, with little algorithmic insight. Saks [33] presented an elegant combinatorial proof by using the product between a poset and a chain of length k, which is

[1] The generalization of Mirsky's results is due to Greene only [21], we use GK to refer to both for simplicity.

[2] We provide a formal definition of k-norm in Section 2.

exploited in the state-of-the-art $(k|E|)^{1+o(1)}$-time solution [27] for MA-k. Later, Frank [17] presented an algorithmic proof based on a reduction to minimum cost flow on the same graph used in Fulkerson's proof [18]. As such, the corresponding algorithm runs on $TC(G)$, making it too slow. However, the structural insights discovered in this proof are exploited in the state-of-the-art $|E|^{1+o(1)}$-time 1/2 approximation [27].

1.1 Overview of our results

In this paper we leverage a less known algorithmic proof of the GK theorems found in Schrijver's book [34, Theorem 14.8 & Theorem 14.10]. These proofs are also based on reductions to minimum cost flow that work directly on G, avoiding the computation of $TC(G)$. We argue that utilizing adaptations of Schrijver's proofs and `greedy` strategies provides a computationally fast and simple framework to solve MCP-k MA-k, MC-k, and MAP-k, by reducing them to a constant number of minimum cost flow computations. Our first results are then obtained by using the recent results on minimum cost flow [11, 36].

▶ **Theorem 1.** *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems MCP-k, MA-k, MC-k and MAP-k in almost optimal $|E|^{1+o(1)}$ time.*

In [27] it was shown how to compute MCP-k [27, Lemma 4.4] and MC-k [27, Lemma 4.6] by a logarithmic number of minimum cost flow computations. The corresponding results in Theorem 1 are simpler and stronger since only one application of minimum cost flow is required, and we additionally solve MA-k in almost optimal time, previously unknown.

Although Theorem 1 solves the problems in optimal time up to subpolynomial factors, it is important to understand whether faster and/or simpler solutions can be achieved by means of parameterization. As mentioned earlier, two state-of-the-art solutions for $k = 1$ (MCP and MA-1) run in time $O(\alpha_1^2|V| + |E|)$ [9, 8] and $\tilde{O}(\alpha_1|E|)$ [30], which is linear and near-linear for constant values of α_1, respectively. It is natural to ask whether such results can be obtained for general k. We obtain the following result.

▶ **Theorem 2.** *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems MCP-k and MA-k in parameterized near linear $\tilde{O}(\alpha_k|E|)$ time and the problems MAP-k and MC-k in parameterized near linear $\tilde{O}(\beta_k|E|)$ time.*

Note that these algorithms are asymptotically faster than the ones in Theorem 1 when $\alpha_k, \beta_k = O(\log^c |E|)$, running in near optimal time. To achieve this, we combine the recent results for negative cost cycles [3, 4, 23, 29] with the simple cycle-canceling method for minimum cost flow. In fact, the minimum cost flow values in Schrijver's reductions are $\alpha_k - |V|$ and $-\beta_k$, respectively. This observation directly derives the result for β_k. To derive the result for α_k, we devise a $\ln |V|$ approximation based on `greedy` for *weighted* set cover (Lemma 15) and prove that the cost of such solution is $\tilde{O}(\alpha_k)$ away from the optimal. As such, this can be seen as a generalization of the simple result in [30], where only the *unweighted* version of `greedy` was used. Analogous to [30], we show how to implement the required `greedy` fast.

Our last results are approximation algorithms based on (unweighted) `greedy` set cover and are independent of Schrijver's reductions. In more detail, it follows from bounds on set cover [25] that the first k `greedy` chains and antichains cover at least a $(1 - 1/e)$ fraction of β_k and α_k, respectively. This observation already beats the state-of-the-art approximation ratio of 1/2 [27]. We show how to implement `greedy` on the set of antichains efficiently.

▶ **Theorem 3.** *Given a DAG $G = (V, E)$ and a positive integer k, there exist $(1 - 1/e)$-approximation algorithms solving MC-k in $O(k|E|)$ time and MA-k in $O(\alpha_1^2|V| + (\alpha_1 + k)|E|)$ time.*

Note that these running times do not depend on α_k, β_k nor have polylog factors as in Theorem 2. Moreover, for constant value of the parameters the algorithms run in optimal time. The result for MC-k is a direct consequence of the algorithm of Mäkinen et al. [30]. For MA-k we show how to compute k `greedy` antichains in $O((\alpha_1 + k)|E|)$ time after using the algorithm of Cáceres et al. [9, 8] to compute the first such antichain. We amortize the computation of the antichains by using the duality between MCP and MA-1 over a subset of vertices. As a by-product, we also obtain a $\ln |V|$ approximation for MAP-k (Lemma 20).

We highlight that the algorithmic results in Theorems 2 and 3 belong to the line of research "FPT inside P" [20, 16, 1, 7, 6, 24] of finding natural parameterizations for problems already in P. We complement our algorithmic contributions with the first (to the best of our knowledge) study of the limitations of `greedy` applied to sets of chains or sets of antichains, as previously exploited in the literature and in this paper. As previously stated, `greedy` inherits the bounds of set cover. Namely, it is a $\ln n$ approximation for the partition problems [12] and a $(1 - 1/e)$ approximation for the coverage problems [25]. It is known that these bound are tight for general set systems [35, 14], however, it remains unknown whether these bounds are tight for sets of chains or antichains of a DAG. We show the following.

▶ **Theorem 4.** *For every $k \geq 2$, there exists a DAG, such that k `greedy` chains cover at most $(1 - 1/e)\beta_k$ vertices. In the case of antichains, k `greedy` antichains cover at most $(3/4)\alpha_k$ vertices.*

We obtain this result by creating DAG instances where `greedy` covers exactly a $(1 - (1 - 1/k)^k)$ fraction of the optimal. For chains, we give a graph for every $k > 1$. For antichains, we give two graphs for $k = 2, 3$. We leave the open problem, whether the antichain case can also be generalized to all $k > 3$. For the partition problems we obtain the following result.

▶ **Theorem 5.** *For MCP, there exists a class of DAGs of increasing size, such that the number of chains taken by `greedy` is a $\log_4(|V|)$ factor away from the optimal $\alpha_1 = 2$. For MAP, the same result holds for `greedy` antichains and $\beta_1 = 2$.*

Although these results do not match the upper bound of $\ln n$, they are the first to demonstrate the approximation ratio to be $\Omega(\log n)$. Moreover, since the results hold for constant $(= 2)$ optimal solution, it also disproves the number of `greedy` chains or antichains to be a function of the optimal solution, that is, α_1 or β_1, respectively. To achieve this, we construct a recursive class of graphs such that, after hiding the vertices chosen by `greedy` in a graph, the resulting graph corresponds to the previous step in the class.

The rest of the paper is organized as follows. In the next section, we present the technical notation used in the paper as well as preliminary results, including Schrijver's reductions. The following four sections are dedicated to prove Theorems 1–5. We prove every theorem in several parts. Missing proofs can be found in the Appendix.

2 Preliminaries and Schrijver's reductions

We use $G = (V, E)$ and k to denote our input DAG and a positive integer, respectively. We denote paths P and chains C (subsequences of paths) by their sequence of vertices, but we also use P and C to refer to the corresponding sets of vertices. As such, their *lengths* are $|P|$ and $|C|$, respectively. Antichains are sets of pairwise unreachable vertices

denoted by A. We use $\mathcal{P}, \mathcal{C}$ and $\mathcal{A}$ to denote collections of paths, chains and antichains, respectively, and $V(\mathcal{P}), V(\mathcal{C}), V(\mathcal{A})$ to denote the corresponding vertices in these collections, e.g. $V(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} C$ and $|V(\mathcal{C})|$ is the *coverage* of $\mathcal{C}$. In the collections of chains $\mathcal{C}$ and antichains $\mathcal{A}$ used in this paper, the chains/antichains are pairwise disjoint. However, pairs of paths in collections of paths $\mathcal{P}$ might intersect. If $v \in A$ (or P or C) we say that A *covers* v, if additionally $A \in \mathcal{A}$ (or $\mathcal{C}$ or $\mathcal{P}$) we also say that $\mathcal{A}$ covers v. If $V = V(\mathcal{P})$ we say that $\mathcal{P}$ is a path *cover*. If $V = V(\mathcal{C})$ (resp. $V(\mathcal{A})$) we say that $\mathcal{C}$ is a chain (resp. antichain) *partition*. The celebrated GK theorems state:

▶ **Theorem 6** (Greene-Kleitman (1976)). *The following two equalities hold:*

$$\alpha_k := \max_{\mathcal{A}:|\mathcal{A}|=k} |V(\mathcal{A})| = \min_{\mathcal{C}:V=V(\mathcal{C})} \sum_{C \in \mathcal{C}} \min(|C|, k)$$

$$\beta_k := \max_{\mathcal{C}:|\mathcal{C}|=k} |V(\mathcal{C})| = \min_{\mathcal{A}:V=V(\mathcal{A})} \sum_{A \in \mathcal{A}} \min(|A|, k)$$

For a chain (or antichain) partition $\mathcal{C}$, $\sum_{C \in \mathcal{C}} \min(|C|, k)$ is known as the k-norm of $\mathcal{C}$. In addition to MC-k and MA-k, we use MP-k to refer to a collection of k paths covering β_k vertices[3]. A path cover of minimum size is denoted MPC. We also use the same notation to refer to the corresponding problems of finding one such optimal collections. See Table 1 for a summary of the acronyms we use. Note that there might be no path cover with k-norm (as defined in Theorem 6) exactly α_k, as paths can be forced to cover vertices multiple times (see e.g. [5, Figure 1]). Schrijver [34] extended the definition of k-norm to collections of paths and (pairwise disjoint) antichains, and proved the following version of the GK theorems. The original result is on sets (paths and antichains) of edges: in Appendix A we show that this can be easily adapted for sets of vertices.

▶ **Theorem 7** (Adaptation of Schrijver (2003)). *The following two equalities hold:*

$$\alpha_k = \min_{\mathcal{P}} |V \setminus V(\mathcal{P})| + k|\mathcal{P}|$$

$$\beta_k = \min_{\mathcal{A}} |V \setminus V(\mathcal{A})| + k|\mathcal{A}|$$

Schrijver used reductions to minimum cost circulation to prove these equalities, we describe these reductions at the end of this section. For a collection of paths (resp. antichains) $\mathcal{P}$, $|V \setminus V(\mathcal{P})| + k|\mathcal{P}|$ is known as the k-norm of $\mathcal{P}$, but we will use Sk-norm to avoid confusion. We use MAS-k (resp. MPS-k) to denote a collection of antichains (resp. paths) of minimum Sk-norm. We define the *partition completion* of a collection of chains (or antichains) $\mathcal{C}$ as $\mathcal{C}^V := \mathcal{C} \cup \{\{v\} \mid v \in V \setminus V(\mathcal{C})\}$.

▶ **Lemma 8.** *Let $\mathcal{C}, \mathcal{A}, \mathcal{P}$ be collections of chains, antichains and paths of a graph $G = (V, E)$, respectively. Then, the following statements hold.*

1. *If $V(\mathcal{P}) = V(\mathcal{C})$, $|\mathcal{C}| = |\mathcal{P}|$ and $|C| \geq k$ for all $C \in \mathcal{C}$, then the Sk-norm of $\mathcal{P}$ equals the k-norm of $\mathcal{C}^V$.*
2. *If $|A| \geq k$ for all $A \in \mathcal{A}$, then the Sk-norm of $\mathcal{A}$ equals the k-norm of $\mathcal{A}^V$.*
3. *If $\mathcal{A}$ is a MAS-k, then $\mathcal{A}^V$ is a MAP-k.*

[3] Note that any MC-k can be converted into a MP-k of the same size and vice versa.

Table 1 Acronyms used for the problems solved in this paper on input $G = (V, E), k$.

Acronym	Definition
MA-k	Maximum Coverage k-antichains, i.e. a set of k antichains whose union size α_k is maximized
MC-k	Maximum Coverage k-chains, i.e. a set of chains whose union size β_k is maximized
MP-k	Maximum Coverage k-paths, i.e., a set of k paths whose union size β_k is maximized
MAP	A partition of V into the minimum number of antichains, which equals β_1, the *height* of G, by Mirsky's theorem [31]
MCP	A partition of V into the minimum number of chains, which equals α_1, the *width* of G, by Dilworth's theorem [13]
MPC	A set of paths covering V of minimum cardinality, which is equal to α_1 by Dilworth's theorem [32]
MAP-k	An antichain partition $\mathcal{A}$ of minimum k-norm $\sum_{A\in\mathcal{A}} \min(\lvert A\rvert, k)$, equaling β_k by Greene-Kleitman theorems [22, 21]; MAP-1=MAP
MCP-k	A chain partition $\mathcal{C}$ of minimum k-norm $\sum_{C\in\mathcal{C}} \min(\lvert C\rvert, k)$, equaling α_k by Greene-Kleitman theorems [22, 21]; MCP-1=MCP
MAS-k	A set $\mathcal{A}$ of antichains of minimum Sk-norm $\lvert V \setminus V(\mathcal{A})\rvert + k\lvert\mathcal{A}\rvert$, equaling β_k by Greene-Kleitman theorems [34]; MAS-k equivalent to MAP-k
MPS-k	A set $\mathcal{P}$ of paths of minimum Sk-norm $\lvert V \setminus V(\mathcal{P})\rvert + k\lvert\mathcal{P}\rvert$, equaling α_k by Greene-Kleitman theorems [34]; MPS-1 equivalent to MPC

Proof.

1. The k-norm of $\mathcal{C}^V$ is

$$\begin{aligned}\sum_{C\in\mathcal{C}^V} \min(|C|,k) &= \sum_{C\in\{\{v\}\mid v\in V\setminus V(\mathcal{C})\}} \min(|C|,k) + \sum_{C\in\mathcal{C}} \min(|C|,k)\\ &= |V\setminus V(\mathcal{C})| + k|\mathcal{C}|\\ &= |V\setminus V(\mathcal{P})| + k|\mathcal{P}|\end{aligned}$$

   And thus equal to the Sk-norm of $\mathcal{P}$.
2. The k-norm of $\mathcal{A}^V$ is

$$\begin{aligned}\sum_{A\in\mathcal{A}^V} \min(|A|,k) &= \sum_{A\in\{\{v\}\mid v\in V\setminus V(\mathcal{A})\}} \min(|A|,k) + \sum_{A\in\mathcal{A}} \min(|A|,k)\\ &= |V\setminus V(\mathcal{A})| + k|\mathcal{A}|\end{aligned}$$

3. It follows by the previous point and Theorems 6 and 7, by noting that $|A| \geq k$ for all $A \in \mathcal{A}$ (removing short, $|A| < k$, antichains decreases the Sk-norm). ◀

Our solutions rely on the use of *minimum flows* and *circulations*, here we review the basics needed to understand our results, for formal definitions we refer to [2, 37]. We say that a digraph $G' = (V', E')$, a demand function $\ell : E' \to \mathbb{Z}_{\geq 0}$, a capacity function $u : E' \to \mathbb{Z}_{\geq 0}$, a cost function $c : E' \to \mathbb{Z}$ and optionally $s \in V'$, $t \in V'$, is a *flow network*. When specifying a flow network, if $\ell(e')$, $u(e')$, or $c(e')$ are not given for some $e' \in E'$, we assume they are $0, \infty, 0$ (∞ being a big enough number), respectively. We say that a flow (or circulation) $f : E' \to \mathbb{Z}_{\geq 0}$ is *feasible* a flow network, if $\ell(e') \leq f(e') \leq u(e')$ for all $e' \in E'$. We denote the value of f by $|f|$: the flow networks used in this paper are always DAGs up to one edge (going from t to s), and thus the value of a feasible flow f is well-defined. Moreover, in all the flow networks considered in this paper, f can be *decomposed* into exactly $|f|$ paths

(removing the edge (t, s) in the case of circulations), that is, f *represents* or *encodes* a path collection $\mathcal{P}_f$ with $|\mathcal{P}_f| = |f|$. There exist algorithms computing $\mathcal{P}_f$ from f in optimal $O(|E'| + \texttt{output})$ time (see e.g [26, 10]). Moreover, Cáceres [5] showed how to compute a chain partition $\mathcal{C}_f$ such that $|\mathcal{C}_f| = |f|$ and $V(\mathcal{P}_f) = V(\mathcal{C}_f)$ in time $O(|E| \log |f|)$. A *minimum flow* (circulation) is a feasible flow minimizing $|f|$. We denote the cost of f by $c(f) = \sum_{e' \in E'} f(e')c(e')$. A *minimum cost flow* (circulation) is a feasible flow minimizing $c(f)$. The problems of minimum flow and minimum cost flow (circulation) have been solved in almost linear $|E|^{1+o(1)}$ time[4] [11, 36].

We denote the *residual graph* of G' and a feasible flow (circulation) f by $G'_f = (V', E'_f)$. Intuitively, for every edge $(u', v') = e' \in E'$, E'_f contains (u', v') with cost $c(e')$ only if $f(e') < u(e')$, and (v', u') with cost $-c(e')$ only if $f(e') > \ell(e')$.

A well-known result from the theory of network flows [37] states that a feasible flow f is a minimum cost circulation if and only if there is no negative cost cycle (the cost of a cycle is the sum its edges' costs) in G'_f. In fact, a negative cost cycle in G'_f can be used to obtain a flow of cost at most $c(f) - 1$. The algorithms in Theorem 2 implement a simple cycle-canceling approach boosted by the recent developments on finding negative cost cycles [3, 4, 23, 29].

▶ **Lemma 9** ([3, 4, 23, 29]). *Let c^* be the minimum cost of a circulation in a flow network G', ℓ, u, c. Given a feasible flow f, we can compute a minimum cost flow in time $\tilde{O}((c(f) - c^* + 1)|E|)$.*

We now present an adaptation of the reduction used by Schrijver in the proof of the GK theorems. For completeness, in Appendix A, we include a complete proof of Theorem 7, adapting the proof of Schrijver to the case of sets (paths and antichains) of vertices.

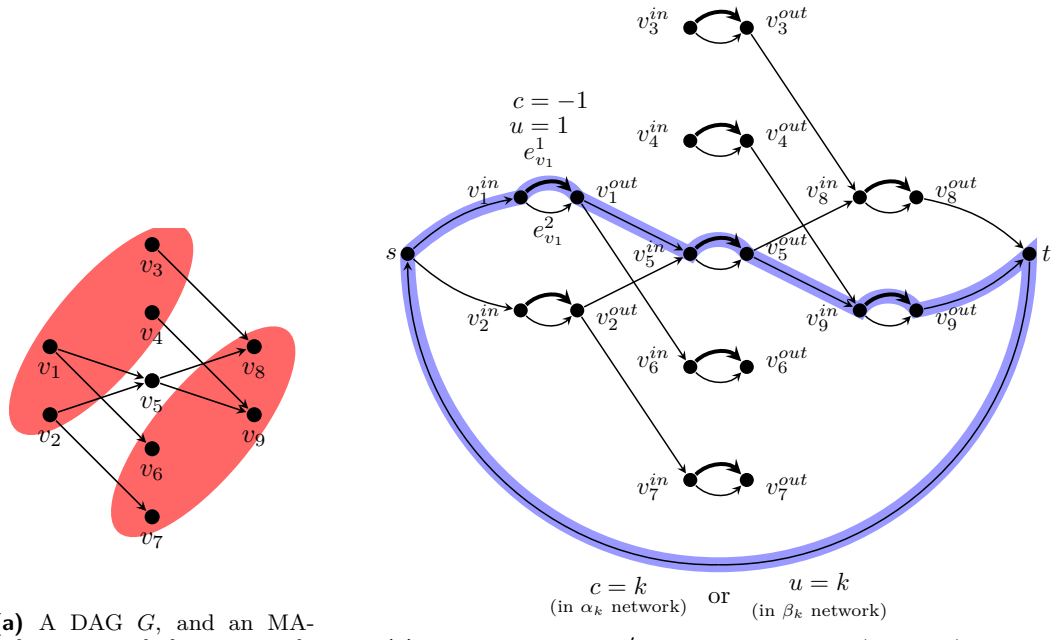
For our inputs $G = (V, E)$ and k, we build the graph $G' = (V', E')$ such that V' contains two vertices v^{in} and v^{out} per vertex $v \in V$ connected by two parallel edges $e^1_v, e^2_v \in E'$ from v^{in} to v^{out}, such that $u(e^1_v) = 1$ and $c(e^1_v) = -1$. This parallel edge construction serves the following purpose. The edge e^1_v acts a "covering edge" that reduces the cost by one and can be used only once, and the edge e^2_v allows us to use vertex v in other paths but does not count towards the covering objective function. For every edge $(u, v) \in E$, E' contains the edge (u^{out}, v^{in}). Additionally, V' contains s and t, with edges $(s, v^{in}), (v^{out}, t) \in E'$ for each $v \in V$. Finally, E' contains the edge (t, s): in the reduction used to prove the α_k part of the theorems $c(t, s) = k$ (in line with Theorem 7), in the reduction used to prove the β_k part $u(t, s) = k$ (allowing at most k paths). We call the two flow networks the α_k *and* β_k *networks*, respectively. Note that the zero circulation is a feasible circulation in both of these networks. See Figure 1 for an example of these networks. Moreover, every minimum circulation f of these networks satisfies $f(e^2_v) \geq 1 \Rightarrow f(e^1_v) = 1$.

In the β_k network, this means that $c(f) = -|V(\mathcal{P}_f)| = -\beta_k$, by Theorem 6. Starting from the zero circulation on Lemma 9, we obtain the following Lemma.

▶ **Lemma 10** (Theorem 2, part II). *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems, MAP-k, MAS-k, MP-k and MC-k in parameterized near linear $\tilde{O}(\beta_k|E|)$ time.*

A feasible circulation f can be decomposed into $\mathcal{P}_f$ and $\mathcal{C}_f$ (to obtain MP-k and MC-k). In Lemma 13 we show how to extract an MAP-k and an MAS-k from the minimum cost circulation. In contrast to the β_k network, in the α_k network we have $c(f) = -|V(\mathcal{P}_f)| + k|\mathcal{P}_f|$

[4] An extra multiplicative factor of $\log U \cdot \log C$ appears in the running time, where U and C are the largest (absolute value of the) capacity and cost, respectively. In the flow networks used in this work, U and C are polynomial in $|E|$ and thus $\log U \cdot \log C = |E|^{o(1)}$.



(a) A DAG G, and an MA-2 $\{v_1, v_2, v_3, v_4\}, \{v_6, v_7, v_8, v_9\}$, in red, covering $\alpha_2 = 8$ vertices.

(b) The flow network G' of G, and an MPS-2 (v_1, v_5, v_9), in blue, of 2-norm equal to $6 + 2 \cdot 1 = 8 = \alpha_2$.

Figure 1 A DAG G and its corresponding α_k and β_k networks G', with their difference on edge (t, s) shown. Namely, in the α_k network, the edge (t, s) gets cost k, while in the β_k network, it gets capacity k. For clarity, most edges of the form (s, v^{in}) and (v^{out}, t) are omitted. The edges of type e^1_v are drawn thicker, and have cost $c = -1$ and capacity $u = 1$.

and its minimum cost is $\alpha_k - |V|$ by Theorem 7. Therefore, starting from a zero circulation, Lemma 9 only produces a $\tilde{O}((\alpha_k - |V| + 1)|E|)$ time algorithm. In Section 4 we show how to efficiently compute a $\ln |V|$ approximation circulation for the α_k network. We obtain the first part of Theorem 2 by starting from such a flow.

3 MA-k in Almost Linear Time

The results in this section follow by connecting previous works. Nevertheless, we believe it is important to present these results, which improve over the state-of-the-art.

We first prove how to obtain Theorem 1 for MPS-k and MCP-k.

▶ **Lemma 11** (Theorem 1, part I). *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems MCP-k and MPS-k in almost optimal $|E|^{1+o(1)} + O(\texttt{output})$ time.*

Proof. We build the α_k network (see Section 2) and compute a minimum cost circulation f in $E^{1+o(1)}$ time [11, 36]. Circulation f can be decomposed into a collection of $|f|$ paths $\mathcal{P}_f$ (for solving MPS-k) or $|f|$ disjoint chains $\mathcal{C}_f$ such that $V(\mathcal{P}_f) = V(\mathcal{C}_f)$ (for solving MCP-k). By construction (see discussion above Lemma 10), the cost of f is $c(f) = -|V(\mathcal{P}_f)| + k|\mathcal{P}_f|$, since f minimizes this cost it also minimizes $|V \setminus V(\mathcal{P})| + k|\mathcal{P}|$, and thus $\mathcal{P}_f$ is a MPS-k. We decompose f into $\mathcal{P}_f$ in $O(|E| + \texttt{output})$ time [26, 10]. We can get $\mathcal{C}_f$ from f in $O(|E| \log |f|) = O(|E| \log |V|) = |E|^{1+o(1)}$ time [5]. In [5, Corollary 6], it is additionally stated that $\mathcal{C}_f$ can be obtained from $\mathcal{P}_f$ by removing the repeated vertices (up to one copy) from the paths of $\mathcal{P}_f$. In other words, there is a one-to-one correspondence between $\mathcal{C}_f$ and $\mathcal{P}_f$ such that $C \in \mathcal{C}_f$ is a subsequence of its corresponding $P \in \mathcal{P}_f$. Next, we note that

for every $P \in \mathcal{P}_f$, $c_P := |P \setminus V(\mathcal{P}_f \setminus \{P\})| \geq k$; indeed, if $c_P < k$ then the Sk-norm of $\mathcal{P}_f \setminus \{P\}$ is $k - c_P$ units smaller than the Sk-norm of $\mathcal{P}_f$, a contradiction. Moreover, since also $V(\mathcal{C}_f) = V(\mathcal{P}_f)$, the corresponding $C \in \mathcal{C}_f$ to P must cover at least those c_P vertices, and thus for every $C \in \mathcal{C}_f, |C| \geq c_P \geq k$. As such, the k-norm of $\mathcal{C}_f^V$ is exactly the Sk-norm of $\mathcal{P}_f$, by Lemma 8, and thus $\mathcal{C}_f^V$ is a MCP-k. ◀

Now we prove how to obtain Theorem 1 for MP-k and MC-k.

▶ **Lemma 12** (Theorem 1, part II). *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems MP-k, MC-k in almost optimal $|E|^{1+o(1)} + O(\texttt{output})$ time.*

Proof. We build the β_k network (see Section 2) and compute a minimum cost circulation f in $E^{1+o(1)}$ time [11, 36]. Circulation f can be decomposed into a collection of $|f|$ paths $\mathcal{P}_f$ (for solving MP-k) or $|f|$ disjoint chains $\mathcal{C}_f$ such that $V(\mathcal{P}_f) = V(\mathcal{C}_f)$ (for solving MC-k). By construction (see discussion in Section 2), the cost of f is $c(f) = -|V(\mathcal{P}_f)|$, since f minimizes this cost, $\mathcal{P}_f$ is a MP-k and $\mathcal{C}_f$ is a MC-k. We decompose f into $\mathcal{P}_f$ in $O(|E| + \texttt{output})$ time [26, 10] and into $\mathcal{C}_f$ in $O(|E| \log |f|) = O(|E| \log |V|) = E^{1+o(1)}$ time [5]. ◀

The proof of Theorem 7 gives an explicit description of the corresponding antichain collections derived from a minimum cost circulation of the α_k and β_k networks. Next, we show how to compute (from the circulation) such antichain collections efficiently. We use this lemma to prove the final part of Theorem 1.

▶ **Lemma 13.** *Given a minimum cost circulation f of the α_k network (resp. β_k network). We can compute a MA-k (resp. a MAS-k and MAP-k) in time $\tilde{O}(|E|)$.*

Proof. Since there are no negative cost cycles in the residual G'_f, the function $d : V' \to \mathbb{Z}$ such that $d(v')$ is the distance of a shortest (minimum cost) path from s to v' in G'_f, is well defined. In fact, we can compute d in $\tilde{O}(|E|)$ by using [3, 4, 23, 29]. The proof of Theorem 7 defines $U_i := \{v' \in V' \mid d(v') \geq i + d(t)\}$ and $A_i = \{v \in V \mid v^{in} \in U'_i \wedge v^{out} \notin U'_i\}$ for every $i \in \{1, \dots d(s) - d(t)\}$. The proof moreover shows that $\mathcal{A}_f = \{A_1, \dots, A_{d(s)-d(t)}\}$ is a collection of antichains and that it is a MA-k in the α_k network and a MAS-k in the β_k network. Note that A_i corresponds to the set of vertices $v \in V$ such that $d(v^{in}) \geq i + d(t)$ and $d(v^{out}) < i + d(t)$. In particular, if $d(v^{in}) > d(v^{out})$, then $v \in A_j$ for $j \in \{d(v^{out}) - d(t) + 1, \dots, d(v^{in}) - d(t)\}$. As such, we can build $\mathcal{A}_f$ in $O(|E|)$ time by checking each edge (v^{in}, v^{out}) and placing v in one such corresponding antichain (e.g. in $A_{d(v^{in})-d(t)}$) when $d(v^{in}) > d(v^{out})$. By Lemma 8, $\mathcal{A}^V$ is a MAP-k. ◀

▶ **Lemma 14** (Theorem 1, part III). *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems MA-k, MAP-k and MAS-k in almost optimal $|E|^{1+o(1)}$ time.*

Proof. Compute a minimum cost circulation f in the α_k network (resp. β_k network) in $|E|^{1+o(1)}$ time [11, 36], and use Lemma 13 to obtain a MA-k (resp. MAS-k and MAP-k). ◀

Proof of Theorem 1. Combine Lemmas 11, 12, and 14. ◀

4 MA-k in Parameterized Near-Linear Time

We exploit the well-known *greedy* algorithm for *weighted set cover* [12]. In *minimum weight set cover* the input is a collection of sets $\mathcal{S}$ such that $\bigcup_{S \in \mathcal{S}} S = V$,[5] and an associated weight function $w : \mathcal{S} \to \mathbb{Z}_{\geq 0}$, and the task is to find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ covering all elements,

[5] The reuse of V in the notation is intentional.

that is $V(\mathcal{S}') = V$, and minimizing its total weight, where $w(\mathcal{S}') = \sum_{S \in \mathcal{S}'} w(S)$. The `greedy` algorithm maintains the set of uncovered elements U and iteratively picks a set S minimizing the ratio $w(S)/|U \cap S|$ until $U = \emptyset$. It is known that `greedy` is a $\ln |V|$ approximation [12]. We show that `greedy` is a $\ln |V|$ approximation for MCP-k and MPS-k, and that it can be implemented efficiently, a generalization of the result of Mäkinen et al. [30, Lemma 2.1].

▶ **Lemma 15.** *We can compute a $\ln |V|$ approximation of MCP-k and MPS-k in time $\tilde{O}(\alpha_k |E|/k)$.*

Proof. We define the following weighted set cover instance: the sets are the chains of G, and for a chain C its weight is $\min(|C|, k)$. Note that a feasible solution for this weighted set cover instance is a chain cover (not necessarily partition), and its weight corresponds to its k-norm. As such, MCP-k's are optimal solutions to this problem. Moreover, `greedy` is a $\ln |V|$ approximation [12]. We next show how to compute such a `greedy` solution efficiently. For this, recall that `greedy` at each step, chooses the chain C minimizing the ratio $w(C)/|U \cap C| = \min(|C|, k)/|U \cap C|$, where U is the set of *uncovered* vertices. First, note that there is always a chain $C \subseteq U$ minimizing this ratio, and thus we assume $C \subseteq U$, and the task becomes to find a chain $C \subseteq U$ minimizing the ratio $\min(|C|, k)/|C|$. Note that this also ensures that the chain collection found by `greedy` is a chain partition. Second, if there is no chain $C \subseteq U$ with $|C| > k$, then the ratio is always one and any such chain minimizes the ratio, when `greedy` reaches this point it can just output every vertex in U in a separated singleton path. Otherwise, `greedy` must find a chain $C \subseteq U$ minimizing $k/|C|$, but since k is a constant for this minimization, it is equivalent to finding a chain $C \subseteq U$ maximizing $|C|$. Such a chain can be obtained by finding a path covering the most vertices from U. Mäkinen et al. [30, Lemma 2.1] show how to compute such a path in $O(|E|)$ time. Since we stop `greedy` as soon as there is no chain $C \subseteq U$ with $|C| > k$, the running time of our approach is the number of chains $|C| > k$ times $O(|E|)$. Finally, since each chain $|C| > k$ contributes k to the k-norm $O(\alpha_k \log |V|)$ of the output cover, the number of such chains is $O(\alpha_k \log |V|/k)$. Note that the Sk-norm of the set of paths and the k-norm of the chain partition output by `greedy` is the same, by Lemma 8. ◀

▶ **Lemma 16** (Theorem 2, part I). *Given a DAG $G = (V, E)$ and a positive integer k, we can solve the problems, MCP-k, MPS-k and MA-k in parameterized near linear $\tilde{O}(\alpha_k |E|)$ time.*

Proof. We use Lemma 15 to compute a collection of paths $\mathcal{P}$ whose Sk-norm is at most $\alpha_k \ln |V|$ in $\tilde{O}(\alpha_k |E|)$ time. These paths correspond to a circulation f in the α_k network. The cost f of this circulation is then $c(f) = -|V(\mathcal{P})| + k|\mathcal{P}| \leq \alpha_k \ln |V| - |V|$, and since the minimum cost of a circulation is $\alpha_k - |V|$ we can use Lemma 9 to obtain a minimum cost circulation in $\tilde{O}(\alpha_k |E|)$ time. As in the proof of Lemma 12, we can obtain a MCP-k and a MPS-k from the minimum cost flow circulation (`output` size for MPS-k is $O(\alpha_k |E|)$). We use Lemma 13 to obtain a MA-k. ◀

Proof of Theorem 2. Combine Lemmas 10 and 16. ◀

5 Approximate MA-k in Parameterized Linear Time

In the *maximum coverage k-sets* version of set cover, the sought solution consists of k sets, covering the maximum number of elements, and in this case `greedy` is stopped after k iterations. It is well-known that this version of `greedy` is a $(1 - 1/e)$ approximation [25]. When applied to the collection of chains/paths or antichains of a DAG, we automatically obtain the approximation ratio claimed in Theorem 3. Therefore, the main purpose of this section is to provide a fast implementation of `greedy` in these contexts.

In a series of works [15, 28, 30] it was shown how to efficiently compute `greedy` chains/paths in $O(|E|)$ time per chain/path, the first part of Theorem 3 follows.

▶ **Lemma 17** (Theorem 3, part I, [30]). *Given a DAG $G = (V, E)$ and a positive integer k, there exist $(1 - 1/e)$-approximation algorithms solving MP-k and MC-k in $O(k|E|)$ time.*

We show how to efficiently compute a maximum-sized antichain $A \subseteq U$ for some subset $U \subseteq V$. We adapt a well-known reduction from MPC to minimum flow [32] to work with minimum path covers only required to cover U instead of the whole V.

▶ **Lemma 18.** *There exists a reduction to minimum flow for the problem of finding a minimum cardinality set of paths covering $U \subseteq V$. Moreover, we can recover a maximum-sized antichain $A \subseteq U$ from a minimum flow in this network in $O(|E|)$ time.*

Proof. For $U = V$, the well-known reduction to minimum flow corresponds to G' as in Schrijver's reduction but without the edge (t, s) and only one copy of edges (v^{in}, v^{out}) for $v \in V$ with $\ell(v^{in}, v^{out}) = 1$, all other (non-specified) values for ℓ and u are set to default (see Section 2). Cáceres et al. [9, 8] show that a minimum flow f in this network corresponds to a MPC of G (by decomposing f) and a maximum-sized antichain can be obtained in $O(|E|)$ time by traversing the residual G'_f from t: the maximum-sized antichain corresponds to the vertices $v \in V$ such that v^{out} is reached by t in G'_f but v^{in} is not. Here, we prove that we obtain the same result when only required to cover $U \subseteq V$ by setting $\ell(v^{in}, v^{out}) = 1$ only for $v \in U$. First, note that a flow in this modified network can be decomposed into a collection of paths covering U and a collection of paths covering U corresponds to a feasible flow in this modified network. As such, a minimum flow f in this modified network can be decomposed into a minimum cardinality set of paths covering $U \subseteq V$. Moreover, consider the vertices V_t reached by t in the residual G'_f of this modified network. We define $A \subseteq U$ as $A = \{v \in U \mid v^{out} \in V_t \wedge v^{in} \notin V_t\}$. First note that, if (by contradiction) there is a vertex $u \in A$ reaching a vertex $v \in A$ in G, then u^{out} reaches v^{in} in G' and also in G'_f (edges have default ∞ capacity), a contradiction since $v^{in} \notin V_t$. As such, A is an antichain. Moreover, by definition, V_t is a ts cut and all edges entering V_t have flow equal to their lower bound. Since there is exactly $|f|$ units of flow entering V_t (by flow conservation), then $|A| = |f|$, and thus A is a maximum-sized antichain $A \subseteq U$. Computing A reduces to computing V_t, which can be done in $O(|E|)$ time. ◀

Therefore, a simple strategy to obtain the k `greedy` antichains is to solve the minimum flow problem defined in Lemma 18 repeatedly, where each time U is defined as the vertices still uncovered. Our final result speeds up this computation by noting that a minimum flow obtained in the i-th step of `greedy` can be used as an initial solution for the step $i + 1$. A well-known result from the theory of network flows [37] states that for a feasible flow f, f is a minimum flow if and only if there is no path from t to s in G'_f. A ts-path in G'_f is known as a *decrementing path* and it is used to obtain a flow of value $\leq |f| - 1$. As such, starting from a feasible flow f, we can obtain a minimum flow f^* by finding $\leq |f| - |f^*|$ decrementing paths in total $O((|f| - |f^*| + 1)|E|)$ time.

▶ **Lemma 19** (Theorem 3, part II). *Given a DAG $G = (V, E)$ and a positive integer k, there exist $(1 - 1/e)$-approximation algorithm solving MA-k in time $O(\alpha_1^2|V| + (\alpha_1 + k)|E|)$.*

Proof. We compute a MPC in time $O(\alpha_1^2|V| + (\alpha_1 + k)|E|)$ [9, 8], compute the corresponding minimum flow f_1 in the reduction of Lemma 18 and obtain the first `greedy` antichain A_1 in $O(|E|)$ time. To obtain the i-th `greedy` antichain A_i, for $i > 1$, we assume that U_i are the vertices still uncovered in iteration i, that is $U_i = V \setminus \bigcup_{j=1}^{i-1} A_j$, and that f_{i-1} is a minimum

flow as in Lemma 18 for U_{i-1}. To compute a minimum flow f_i for U_i we use f_{i-1} as an initial solution. Note that f_{i-1} is a feasible flow in the network for U_i since $U_i \subseteq U_{i-1}$. Obtaining f_i from f_{i-1}, can be done by finding $|f_{i-1}| - |f_i|$ decrementing ts paths in the corresponding residual networks, each in $O(|E|)$ time. Once we obtain f_i, we can obtain A_i (and U_i) in $O(|E|)$ time by the second part of Lemma 18. The total running time after computing the MPC is then $O\left(|E| \sum_{i=2}^{k} |f_{i-1}| - |f_i| + 1\right) = O(|E|(k + |f_1| - |f_k|)) = O((\alpha_1 + k)|E|)$. ◀

Proof of Theorem 3. Combine Lemmas 17 and 19. ◀

Combining the ideas from Lemmas 15 and 19 we obtain the following.

▶ **Lemma 20.** *We compute a* $\ln |V|$ *approximation of MAS-k and MAP-k in* $\tilde{O}((\alpha_1 + \beta_k/k)|E|)$ *time.*

Proof. We start by computing an MPC in $\tilde{O}(\alpha_1|E|)$ time [30], and compute `greedy` antichains as in the proof of Lemma 19 but stopping the computation whenever the next antichain size is at most k (as we did for chains/paths in the proof of Lemma 15) obtaining a collection of `greedy` antichains: $\mathcal{A}$. Note that, following the analysis of Lemma 19, the running time of computing $\mathcal{A}$ is $O((\alpha_1 + |\mathcal{A}|)|E|)$. We next prove that $\mathcal{A}$ is a $\ln |V|$ approximation for the MAS-k problem. The bound $|\mathcal{A}| \leq \beta_k \ln |V|/k$ follows by noting that each antichain in $\mathcal{A}$ contributes exactly k units to its Sk-norm. The fact that $\mathcal{A}$ is a $\ln |V|$ approximation follows by noting, as in the proof of Lemma 15, that the implementation `greedy` over the set of antichains of G with weights $\min(|A|, k)$ for antichain A, is exactly the first antichains of (unweighted) `greedy` until their size becomes at most k (for MAS-k) plus singleton antichains for the yet uncovered vertices (for MAP-k). The Sk-norm of the antichain collection (for MAS-k) is equal to the k-norm of the antichain partition (for MAP-k) by Lemma 8. ◀

6 Limitations of Greedy on Chains/Paths and Antichains

▶ **Lemma 21** (Theorem 4, part I). *For every* $k \geq 2$*, there exists a DAG, such that* k `greedy` *chains cover at most* $(1 - 1/e)\beta_k$ *vertices in the worst case, which is tight.*

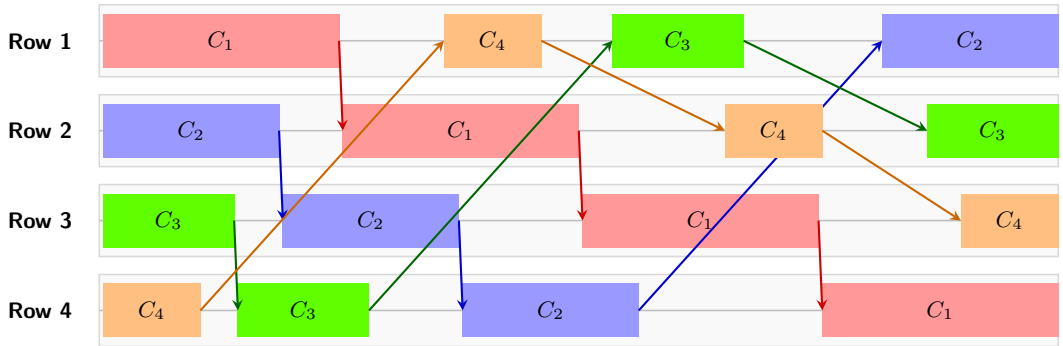
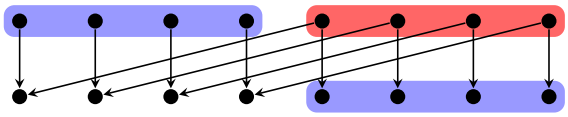
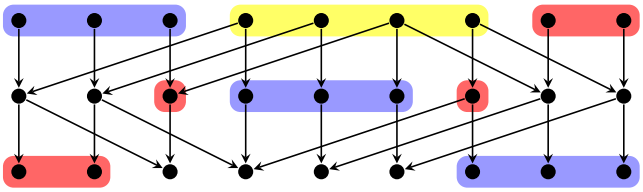


Figure 2 Visualization of the graph G_k in Lemma 21. Colored blocks represent segments of the `greedy` paths where C_i is the i^{th} path that `greedy` takes. For example, using the definition in the proof of Lemma 21, $\Gamma_{2,3} = \{49, \ldots, 96\}$ corresponds to the blue block C_2 in row 3.

Proof sketch. We construct a class of graphs G_k for $k \in \mathbb{N}$ and show that `greedy` covers at most a $1 - (1 - 1/k)^k$-ratio of vertices of G_k, while $\beta_k = |V|$. We introduce k^{k+1} vertices that can be covered by k paths of each k^k vertices. This set of paths represents the optimal



(a) The graph G_2 from Lemma 22.



(b) The graph G_3 from Lemma 22.

Figure 3 Illustrations of the graphs used in Lemma 22. The `greedy` antichains are obtained in the order blue, red, yellow.

solution and we call the i-th such path the i-th row of the graph (see Figure 2). We label the vertices accordingly: v_i^j with $i \in [k]$, $j \in [k^k]$ denotes the j-th vertex in the i-th row. The idea is that `greedy` will cover a $1/k$ fraction of the remaining uncovered vertices in each row.

The set of vertices $C_i \subseteq V(G_k)$ to be covered by the i-th `greedy` chain ($i \in [k]$) is defined as follows: $C_i = \{v_j^\ell \mid j \in [k], \ell \in \Gamma_{i,j}\}$, where the index set $\Gamma_{i,j}$ is defined as the integers inside the following interval:

$$\Gamma_{i,j} = \mathbb{N} \cap \begin{cases} [(j-i)(1-\frac{1}{k})^{i-1}k^{k-1}+1,\ (j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1}] & \text{if } j \geq i, \\ [(k+j-i)(1-\frac{1}{k})^{i-1}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1}+1, & \\ \quad (k+j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1}] & \text{if } j < i. \end{cases}$$

We add the edges to G_k that connect consecutive vertices v_j^ℓ in C_i sorted in ascending order by ℓ. Informally, the chains are paths in G_k that look like stairs, where the i-th `greedy` chain starts on the first vertex of row i and loops from row k to row 1 if $i > 1$. Clearly, G_k is a DAG, because all edges have the form $(v_{i_1}^{j_1}, v_{i_2}^{j_2})$ for some $i_1, i_2 \in [k]$ and $j_2 > j_1$. We will show the following Properties:

1. The sets C_i are well-defined and pairwise disjoint,
2. The set C_i is of size $(1-\frac{1}{k})^{i-1}k^k$ and covers $(1-\frac{1}{k})^{i-1}k^{k-1}$ vertices in each row,
3. The i-th `greedy` path covers precisely the set C_i in the worst case.

Showing these three Properties concludes the proof, as the number of paths chosen by `greedy` divided by the optimal solution is the following:

$$\frac{1}{\beta_k}\left|\bigcup_{i\in[k]} C_i\right| = \frac{1}{\beta_k}\sum_{i=1}^{k}|C_i| = \frac{1}{k}\sum_{i=1}^{k}\left(1-\frac{1}{k}\right)^{i-1} \xrightarrow{k\to\infty} \left(1-\frac{1}{e}\right).$$

We finish the proof by showing Properties 1-3 in Appendix B. ◀

▶ **Lemma 22** (Theorem 4, part II). *For every $k \geq 2$, there exists a DAG, such that k `greedy` antichains cover at most $(3/4)\alpha_k$ vertices in the worst case.*

Proof. We divide this proof into two parts: first, we give two instances, for $k = 2$ and $k = 3$, of DAGs such that k `greedy` antichains cover at most $(3/4)\alpha_k$ vertices. Secondly, we combine these two instances to create an instance for any k. Let G_2 be an instance for $k = 2$

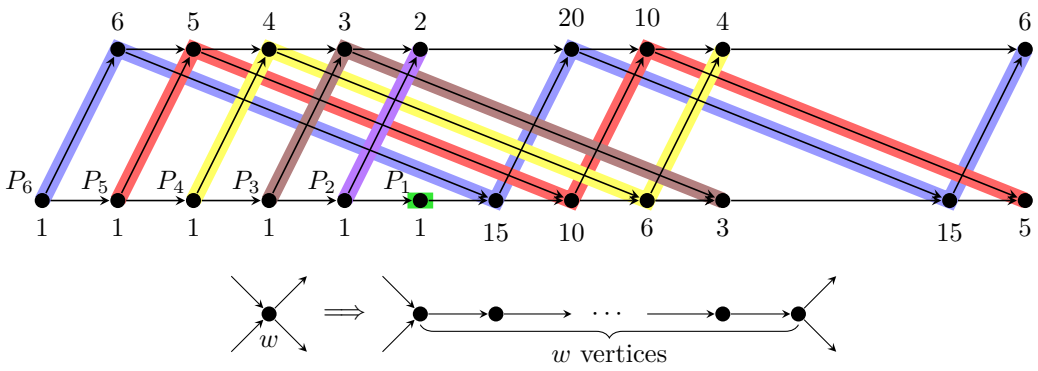


Figure 4 Illustration of the instance G_6^C, with 6 alternating paths shows as different colors. Each vertex is labeled by its weight w (which can be reduced to w vertices as shown in the bottom of the figure). The green path of a single vertex is P_1, and the blue path of length 6 is P_6. The vertices are ordered by $P_6[j-1] \to P_5[j-1] \to \cdots \to P_j[j-1]$ for every $j = 1, \ldots, 6$. Moreover, we ensure that $P_j[j-1] \to P_6[j+1]$ for all $j = 1, \ldots, 4 = 6-2$.

and G_3 be an instance where $k = 3$. See Figure 3 for G_2 and G_3. In the G_i instance, for $i = 2, 3$, $\alpha_i = |V(G_i)|$ by selecting i rows. `greedy`, however, covers $8 + 4 = 12$ vertices in G_2 by selecting the blue and red antichains in this order, and covers $9 + 6 + 4 = 19$ vertices in G_3 by selecting the blue, red, yellow antichains, in this order. This gives us the ratio of $12/16 = 3/4$ for $k = 2$ and $19/27 < 3/4$ for $k = 3$.

To obtain the 3/4 bound for even k, we use $\frac{k}{2}$ copies of G_2 (called $G_2^{(1)}, G_2^{(2)}, \ldots, G_2^{(\frac{k}{2})}$). Then we add edges from each vertex in $G_2^{(i)}$ to each vertex in $G_2^{(i+1)}$ for all i. These edges make sure that no antichain contains vertices from different $G_2^{(i)}$. `Greedy` covers 3/4 portion of vertices of each G_2. For odd k, we use $\frac{k-1}{2}$ copies of G_2 (called $G_2^{(1)}, G_2^{(2)}, \ldots, G_2^{(\frac{k-1}{2})}$) and one copy of G_3. Then we again add edges from each vertex in $G_2^{(i)}$ to each vertex in $G_2^{(i+1)}$ for all i, and also from each vertex in $G_2^{(\frac{k-1}{2})}$ to each vertex in G_3. `Greedy` covers at most 3/4 portion of vertices of G_3 and each G_2, this gives us the final ratio of 3/4. ◀

Proof of Theorem 4. Combine Lemmas 21 and 22. ◀

We now prove the first part of Theorem 5 by providing a class of graphs such that `greedy` a $\log_4(|V|)$ factor away from the optimal $\alpha_1 = 2$ for the problems MPC and MCP. We recursively define a class of DAGs G_i^C for $i \geq 2$, each of which can be covered by two paths.

See Figure 4 for an illustration of the DAGs $G_i^C = (V_i^C, E_i^C)$. We use *weights* w as a shorthand notation for a path of w vertices. Notice that the optimal solution has size 2 by taking top and bottom paths (we call these *straight paths*). We then add paths with vertices that `greedy` will take (we call these *alternating paths*). The graph G_i^C has i alternating paths $P_1, \ldots, P_i$, which always start from the bottom straight path. The number of times that P_i alternates between the top and bottom paths is $i - 1$. We use the notation $P_i = \{P_i[0], \ldots, P_i[i]\}$ separated by each alternation. The base case graph G_1^C consists of one alternating path P_1 consisting of a single node with weight 1. The i-th instance G_i^C is constructed by adding the alternating path P_i whose weights $w(P_i[j]) = \binom{i}{j}$. The vertices of the paths are chosen to respect the following order: $P_i[j-1] \to P_{i-1}[j-1] \to \cdots \to P_j[j-1]$ for all $j = 1, \ldots, i$ and, in addition, $P_j[j-1] \to P_i[j+1]$ for all $j = 1, \ldots, i-2$.

▶ **Lemma 23** (Theorem 5, part I). *For MPC and MCP, the number of chains/paths taken by* `greedy` *on the instances* G_i^C *is a* $\log_4(|V|)$ *factor away from the optimal* $\alpha_1 = 2$.

Proof. `Greedy` returns equivalent solutions for MPC and MCP. We show the following:

1. In G_i^C, `greedy` chooses the alternating path P_i in its first iteration,
2. $TC(G_i^C) \setminus P_i = TC(G_{i-1}^C)$ for all $i > 1$.

The second statement implies that after removing the vertices of P_i from $TC(G_i^C)$[6], the set of the remaining edges corresponds to exactly all the paths in G_{i-1}^C. As a result, we can recursively use Statement 1 and 2, to show that the paths P_i are each taken by `greedy`.

In the unweighted version of G_i^C, the number of vertices is the sum of all the weights. That is, $w(V_i^C) = |V_i^C| = \sum_{j=1}^{i} |P_j| = \sum_{j=1}^{i} 2^j - 1 < 2^{i+1}$, which implies that the number of `greedy` paths is $i \geq \log_2(|V_i^C|)$. Dividing by $\alpha_1 = 2$, we obtain $\log_4(|V_i^C|)$ for the factor.

It remains to show both statements. We first show Statement 1. Let $i \geq 2$ and let $Q = \{Q[0], \ldots, Q[|Q|-1]\}$ be the first path chosen by `greedy`. We will show that P_i is the unique largest weight path in G_i^C and thus, $Q = P_i$. To observe this, we first note that $P_i[0]$ is the unique source node of the DAG, which implies that $Q[0] = P_i[0]$. Assume that $Q \neq P_i$. We will locally change Q until $Q = P_i$ and strictly increase its weight in the process. Let j be the smallest index of Q such that $Q[j] \neq P_i[j]$

- If $j = i-1$ ($P_i[j]$ last vertex of P_i), since P_i is the longest alternating path, $|Q| = i$ and $Q[j] = P_{i-1}[j-1]$. We reroute Q to $P_i[j]$, thus $w(P_{i-1}[j-1]) = \binom{i-1}{j-1} < \binom{i}{j} = w(P_i[j])$.
- If $j < i-1$, we reroute Q by passing it through $P_i[j]$, and reconnect it again with a later vertex from Q in the following way:
  - If Q passes through $P_i[j+1]$, it did not traverse an alternating edge, and we reroute Q through $P_i[j]$ by using the alternating edge to $P_i[j+1]$. In this case, we replace

    $$Q = \{P_i[0], \ldots, P_i[j-1], \mathbf{P_{i-1}[j-1]}, \ldots, \mathbf{P_j[j-1]}, P_i[j+1], \ldots\}$$

    by

    $$Q = \{P_i[0], \ldots, P_i[j-1], \mathbf{P_i[j]}, P_i[j+1], \ldots\}.$$

    We have

    $$w(P_{i-1}[j-1]) + \cdots + w(P_j[j-1]) = \binom{i-1}{j-1} + \cdots + \binom{j}{j-1} < \binom{i}{j} = w(P_i[j]).$$

  - If Q does not pass through $P_i[j+1]$, we reroute Q through $P_i[j]$ using the unique straight path to reconnect to Q. In this case, we replace

    $$Q = \{P_i[0], \ldots, P_i[j-1], \mathbf{P_{i-1}[j-1]}, \ldots, \mathbf{P_{i-\ell}[j-1]}, P_{i-\ell}[j], \ldots\}$$

    where $\ell \leq i-j+1$, by

    $$Q = \{P_i[0], \ldots, P_i[j-1], \mathbf{P_i[j]}, \mathbf{P_{i-1}[j]}, \ldots, P_{i-\ell}[j], \ldots\}.$$

    We have

    $$w(P_{i-1}[j-1]) + \cdots + w(P_{i-\ell}[j-1]) = \binom{i-1}{j-1} + \cdots + \binom{i-\ell}{j-1} < \binom{i}{j} = w(P_i[j]).$$

[6] In the DAG G vertices can be traversed by `greedy` multiple times in order to reach uncovered vertices, but in the transitive closure $TC(G)$, this is not necessary.

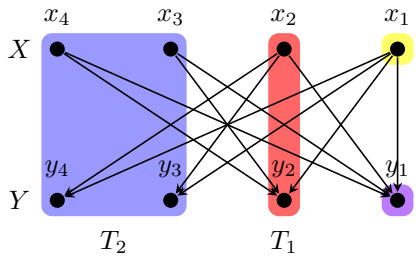


Figure 5 Illustration of the graph G_2^A from Lemma 24.

We proceed with growing index j until we reach the last vertex of P_i. The weight of Q strictly increased after every reroute, and we thus showed that P_i is the unique largest weight path in G_i^C. Next, we show Statement 2. Clearly, $TC(G_{i-1}^C) \subseteq TC(G_i^C) \setminus P_i$. To show the opposite inclusion, let $u, v \in V_{i-1}^C$, such that the edge (u, v) is in $TC(G_i^C) \setminus P_i$. Let $u = P_j[\ell]$ and $v = P_k[\ell']$, where $j, k \leq i-1$. We show that there exists a uv-path in G_{i-1}^C, which shows that the edge (u, v) is also in $TC(G_{i-1}^C)$. Assume that u and v are covered by different straight paths (otherwise, the straight path is the uv-path we are looking for). Then we can consider either $u' = P_j[\ell+1]$ if $j > \ell+1$, or $u' = P_{i-1}[\ell+3]$ if $j = \ell+1$. In both cases, u' is the left most vertex on the same straight path as v satisfying that (u, u') is an edge in $TC(G_{i-1}^C)$. Thus, also (u', v) is an edge in $TC(G_{i-1}^C)$. ◀

▶ **Lemma 24** (Theorem 5, part II). *For MAP, there exists a class of DAGs of increasing size, such that the number of antichains taken by* `greedy` *is a* $\log_4(|V|)$ *factor away from the optimal* $\beta_1 = 2$ *in the worst case.*

Proof. We construct a class of instances $G_i^A = (V_i^A, E_i^A)$ such that $|V_i^A| = 2 \cdot 2^i$, $\beta_1 = 2$ and `greedy` takes $i+2$ antichains. This implies the factor of $(i+2)/2 \geq \log_4(|V_i^A|)$. See Figure 5 for an illustration. The undirected graph underlying the DAG G_i^A is bipartite, with $V_i^A = X \cup Y$, $X = \{x_1, \ldots, x_{2^i}\}$ and $Y = \{y_1, \ldots, y_{2^i}\}$. Since the graph is bipartite, X and Y are both antichains that correspond to the optimal solution. We add edges from X to Y, such that the following sets become antichains for $1 \leq k \leq i$: $T_k = \{x_j, y_j \mid 2^{k-1} < j \leq 2^k\}$. We let $(x_j, y_{j'}) \in E_i^A$ if and only if they are not both in the same set T_k (i.e., $|T_k \cap \{x_j, y_{j'}\}| \leq 1$ for all $1 \leq k \leq i$). Note that if a non-singleton set $U \subseteq V_i^A$ is an antichain, then it is a subset of some T_k, X or Y. It can easily be verified by induction that `greedy` chooses the sets T_k, from $k = i$ to 1, with two singleton vertices x_1 and y_1 left over, that are chosen as two new antichains, hence $i+2$ antichains in total. ◀

Note that `greedy`'s worst case solutions are not unique in Lemma 21, Lemma 22 and Lemma 24, however, they are unique in Lemma 23.

Proof of Theorem 5. Combine Lemmas 23 and 24. ◀

References

1 Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.CH28.

2 Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Teory, Algorithms, and Applications*. Prentice-Hall, 1993.

3 Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *IEEE 63rd annual symposium on foundations of computer science (FOCS 2022)*, pages 600–611. IEEE, 2022. doi:10.1109/FOCS54457.2022.00063.

4 Karl Bringmann, Alejandro Cassis, and Nick Fischer. Negative-weight single-source shortest paths in near-linear time: Now faster! In *IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 515–538. IEEE, 2023. doi:10.1109/FOCS57990.2023.00038.

5 Manuel Cáceres. Minimum Chain Cover in Almost Linear Time. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.31.

6 Manuel Cáceres. Parameterized algorithms for string matching to DAGs: Funnels and beyond. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPIcs*, pages 7:1–7:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CPM.2023.7.

7 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 *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 257–269. Springer, 2021. doi:10.1007/978-3-030-86838-3_20.

8 Manuel Caceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I Tomescu. Minimum path cover in parameterized linear time. *arXiv preprint arXiv:2211.09659*, 2022. doi:10.48550/arXiv.2211.09659.

9 Manuel Cáceres, Massimo Cairo, Brendan Mumey, Romeo Rizzi, and Alexandru I Tomescu. Sparsifying, shrinking and splicing for minimum path cover in parameterized linear time. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 359–376. SIAM, 2022. doi:10.1137/1.9781611977073.18.

10 Manuel Cáceres, Brendan Mumey, Santeri Toivonen, and Alexandru I Tomescu. Practical minimum path cover. In *International Symposium on Experimental Algorithms (SEA 2024)*, pages 1–19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SEA.2024.3.

11 Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

12 Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

13 Robert P Dilworth. A decomposition theorem for partially ordered sets. *Classic Papers in Combinatorics*, pages 139–144, 1987.

14 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

15 Stefan Felsner, Vijay Raghavan, and Jeremy Spinrad. Recognition algorithms for orders of small width and graphs of small dilworth number. *Order*, 20:351–364, 2003. doi:10.1023/B:ORDE.0000034609.99940.FB.

16 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms (TALG)*, 14(3):1–45, 2018. doi:10.1145/3186898.

17 András Frank. On chain and antichain families of a partially ordered set. *Journal of Combinatorial Theory, Series B*, 29(2):176–184, 1980. doi:10.1016/0095-8956(80)90079-9.

18 Delbert R Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. *Proceedings of the American Mathematical Society*, 7(4):701–702, 1956.

19 Fănică Gavril. Algorithms for maximum k-colorings and k-coverings of transitive graphs. *Networks*, 17(4):465–470, 1987. doi:10.1002/NET.3230170407.

20 Archontia C Giannopoulou, George B Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *Theoretical Computer Science*, 689:67–95, 2017. doi:10.1016/J.TCS.2017.05.017.

21 Curtis Greene. Some partitions associated with a partially ordered set. *Journal of Combinatorial Theory, Series A*, 20(1):69–79, 1976. doi:10.1016/0097-3165(76)90078-9.

22 Curtis Greene and Daniel J Kleitman. The structure of Sperner k-families. *Journal of Combinatorial Theory, Series A*, 20(1):41–68, 1976. doi:10.1016/0097-3165(76)90077-7.

23 Bernhard Haeupler, Yonggang Jiang, and Thatchaphol Saranurak. Deterministic negative-weight shortest paths in nearly linear time via path covers. *arXiv preprint arXiv:2511.08551*, 2025. doi:10.48550/arXiv.2511.08551.

24 Anne-Sophie Himmel, George B Mertzios, André Nichterlein, and Rolf Niedermeier. Fast parameterized preprocessing for polynomial-time solvable graph problems. *Communications of the ACM*, 67(4):70–79, 2024. doi:10.1145/3624713.

25 Dorit S Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143. PWS Publishing Company, 1996.

26 Shimon Kogan and Merav Parter. Beating Matrix Multiplication for $n^{1/3}$-Directed Shortcuts. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.82.

27 Shimon Kogan and Merav Parter. The Algorithmic Power of the Greene-Kleitman Theorem. In *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 80:1–80:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.80.

28 Mirosław Kowaluk, Andrzej Lingas, and Johannes Nowak. A path cover technique for LCAs in dags. In *Scandinavian Workshop on Algorithm Theory*, pages 222–233. Springer, 2008. doi:10.1007/978-3-540-69903-3_21.

29 Jason Li. Deterministic padded decompositions and negative-weight shortest paths. *arXiv preprint arXiv:2511.07859*, 2025. doi:10.48550/arXiv.2511.07859.

30 Veli Mäkinen, Alexandru I Tomescu, Anna Kuosmanen, Topi Paavilainen, Travis Gagie, and Rayan Chikhi. Sparse dynamic programming on DAGs with small width. *ACM Transactions on Algorithms*, 15(2):1–21, 2019. doi:10.1145/3301312.

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

32 Simeon C Ntafos and S Louis Hakimi. On path cover problems in digraphs and applications to program testing. *IEEE Transactions on Software Engineering*, 5(5):520–529, 1979. doi:10.1109/TSE.1979.234213.

33 Michael Saks. A short proof of the existence of k-saturated partitions of partially ordered sets. *Advances in Mathematics*, 33(3):207–211, 1979.

34 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

35 Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC 1996)*, pages 435–441, 1996. doi:10.1145/237814.237991.

36 Jan Van Den Brand, Li Chen, Richard Peng, Rasmus Kyng, Yang P Liu, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. A deterministic almost-linear time algorithm for minimum-cost flow. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 503–514. IEEE, 2023. doi:10.1109/FOCS57990.2023.00037.

37 David P Williamson. *Network flow algorithms*. Cambridge University Press, 2019.

A Proof of the Greene-Kleitman Theorems

▶ **Theorem 7** (Adaptation of Schrijver (2003)). *The following two equalities hold:*

$$\alpha_k = \min_{\mathcal{P}} |V \setminus V(\mathcal{P})| + k|\mathcal{P}|$$
$$\beta_k = \min_{\mathcal{A}} |V \setminus V(\mathcal{A})| + k|\mathcal{A}|$$

Proof. This proof mimics the proofs of [34, Theorem 14.8 & Theorem 14.10] for the case of paths/antichains of vertices. We first prove the inequality $\leq$, which holds for any k antichains/k paths and any path/antichain collection. For the α_k version, let $\mathcal{A}$ be k antichains, $|\mathcal{A}| = k$, and $\mathcal{P}$ a collection of paths. Then, note that

$$\begin{aligned}|V(\mathcal{A})| &\leq |V \setminus V(\mathcal{P})| + |V(\mathcal{P}) \cap V(\mathcal{A})| \\ &\leq |V \setminus V(\mathcal{P})| + \sum_{P \in \mathcal{P}} \sum_{A \in \mathcal{A}} |P \cap A| \\ &\leq |V \setminus V(\mathcal{P})| + k|\mathcal{P}|\end{aligned}$$

where the last inequality follows since a path and an antichain can intersect in at most one vertex, i.e., $|P \cap A| \leq 1$. The β_k version of the inequality follows by the same argument (by exchanging the roles of $\mathcal{P}$ and $\mathcal{A}$). To prove that the values meet in the optimum we use the α_k and β_k networks described in Section 2. Consider a minimum cost circulation f in one of the networks. As argued in the main paper, if f is minimum in the α_k network, then we can obtain a MPS-k $\mathcal{P}_f$ by decomposing f, and if f is minimum in the β_k network, then we can obtain a MP-k $\mathcal{P}_f$ by decomposing f. In both networks, consider the function $d : V' \to \mathbb{Z}$ such that $d(v')$ is the distance of a shortest (minimum cost) path from s to v' in the residual G'_f. First, note that since f is minimum, there are not negative cost cycles in G'_f and thus d is well defined. Moreover, $d(s) = 0$ by definition, and $d(v') \leq 0$ for all $v' \in V'$ since there is always a path $P = s, v^{in}, v^{out}, t$ (using e^2_v) in G'_f such that all edge costs are 0 and $v' \in P$. Furthermore, since d is the shortest path distance from s in G'_f, it follows that for all $e' = (u', v') \in E'_f$, $d(v') \leq d(u') + c(e')$. In particular, if both an edge $e' = (u', v')$ and its reverse are in E'_f, then $d(v') = d(u') + c(e')$. For our networks, this implies:

1. $d(v') \leq d(u')$ if $e' = (u', v') \in E' \setminus (\{e^1_v \mid v \in V\} \cup \{(t, s)\})$, and $d(v') = d(u')$ if $f(e') \geq 1$. Cost in such edges is $c(e') = 0$ and its reverse is in E'_f if and only if $f(e') \geq 1$.
2. $d(v^{out}) \leq d(v^{in}) - 1$ if $f(e^1_v) = 0$, and $d(v^{out}) \geq d(v^{in}) - 1$ if $f(e^1_v) = 1$. Cost in such edges is $c(e^1_v) = -1$ and $e^1_v \in E'_f \Leftrightarrow f(e^1_v) = 0 \Leftrightarrow$ reverse of $e^1_v \notin E'_f$.
3. For the α_k network, $d(s) = d(t) + k$, since we assume $f(t, s) \geq 1$: otherwise, there is no path of length more than k (one such path would reduce $c(f)$) and by Mirsky's theorem [31] a MAP is a MA-k whose coverage ($\alpha_k = |V|$) matches the Sk-norm of $\mathcal{P}_f = \emptyset$.
4. For the β_k network, we assume $f(t, s) = k$: otherwise we can redefine f to be a different minimum cost circulation with the same cost and $f(t, s) = k$ (e.g. pick arbitrary $v \in V$ and push $k - f(t, s)$ units of flow on $(s, v^{in}), e^2_v, (v^{out}, t)$ and (t, s) at total cost 0).

We define the following subset of vertices $U_i := \{v' \in V' \mid d(v') \geq i + d(t)\}$ for $i \in \{1, \ldots, d(s) - d(t)\}$. By definition, U'_i is an st-cut ($s \in U'_i$, $t \notin U'_i$). Moreover, there are no edges, other than (t, s) going into U'_i in G': indeed, if $(u', v') \in E' \setminus \{(t, s)\}, u' \notin U'_i, v' \in U'_i$, then by **1.** and **2.**, $d(v') \leq d(u')$, but also $d(u') < i + d(t)$ and $d(v') \geq i + d(t)$ implying $d(u') < d(v')$, a contradiction. We use U_i to define subsets of vertices of the input DAG G, $A_i = \{v \in V \mid v^{in} \in U'_i \wedge v^{out} \notin U'_i\}$. In fact, A_i is an antichain: if $u \neq v \in A$ are such that

there is a uv path in G, then there is a $u^{out}v^{in}$ path in $G' \setminus \{(t,s)\}$, but such a path would cross from $V' \setminus U_i$ to U_i, a contradiction. We define $\mathcal{A}_f = \{A_1, \ldots, A_{d(s)-d(t)}\}$ to be the collection of antichains extracted from f. Note that in the α_k network, by **3.**, $|\mathcal{A}_f| = k$. And in the β_k network, by **4.**, $|\mathcal{P}_f| = k$. These collections satisfy:

a. $V \setminus V(\mathcal{P}_f) \subseteq V(\mathcal{A}_f)$: indeed, if $v \in V \setminus V(\mathcal{P}_f)$, then $f(e_v^1) = 0$ and, by **2.**, $d(v^{out}) \leq d(v^{in}) - 1$. But then, $v \in A_{d(v^{in})-d(t)} \subseteq V(\mathcal{A}_f)$. Note that, $1 \leq d(v^{out}) + 1 - d(t) \leq d(v^{in}) - d(t) \leq d(s) - d(t)$ since both $(s, v^{in}), (v^{out}, t) \in E'_f$.

b. Thus also $V \setminus V(\mathcal{A}_f) \subseteq V(\mathcal{P}_f)$.

For the α_k network we have:

$$\begin{aligned}
k|\mathcal{P}_f| &= \\
\text{(by } \mathbf{3.}) &= (d(s) - d(t))f(t,s) \\
(f \text{ is circulation}) &= \sum_{e'=(u',v') \in E' \setminus \{(t,s)\}} (d(u') - d(v'))f(e') \\
&= \sum_{e'=(u',v') \in E' \setminus (\{e_v^1 | v \in V\} \cup \{(t,s)\})\})} (d(u') - d(v'))f(e') \\
&\quad + \sum_{e_v^1, v \in V} (d(v^{in}) - d(v^{out}))f(e_v^1) \\
\text{(by } \mathbf{1.}) &= \sum_{e_v^1, v \in V} (d(v^{in}) - d(v^{out}))f(e_v^1) \\
\text{(by } \mathbf{2.}) &= |\{v \in V \mid f(e_v^1) = 1 \wedge d(v^{in}) - d(v^{out}) = 1\}| \\
&\leq |V(\mathcal{P}_f) \cap V(\mathcal{A}_f)|
\end{aligned}$$

For the β_k network we have:

$$\begin{aligned}
k|\mathcal{A}_f| &= \\
\text{(by } \mathbf{4.}) &= f(t,s)(d(s) - d(t)) \\
\text{(previous derivation)} &\leq |V(\mathcal{P}_f) \cap V(\mathcal{A}_f)|
\end{aligned}$$

For α_k the proof concludes by:

$$\begin{aligned}
|V \setminus V(\mathcal{P}_f)| + k|\mathcal{P}_f| &= \\
\text{(by } \mathbf{a.}) &= |V(\mathcal{A}_f) \setminus V(\mathcal{P}_f)| + k|\mathcal{P}_f| \\
\text{(derivation for } \alpha_k \text{ network)} &\leq |V(\mathcal{A}_f) \setminus V(\mathcal{P}_f)| + |V(\mathcal{P}_f) \cap V(\mathcal{A}_f)| \\
&= |V(\mathcal{A}_f)|
\end{aligned}$$

Analogously for β_k:

$$\begin{aligned}
|V \setminus V(\mathcal{A}_f)| + k|\mathcal{A}_f| &= \\
\text{(by } \mathbf{b.}) &= |V(\mathcal{P}_f) \setminus V(\mathcal{A}_f)| + k|\mathcal{A}_f| \\
\text{(derivation for } \beta_k \text{ network)} &\leq |V(\mathcal{P}_f) \setminus V(\mathcal{A}_f)| + |V(\mathcal{P}_f) \cap V(\mathcal{A}_f)| \\
&= |V(\mathcal{P}_f)|
\end{aligned}$$

◀

B Omitted proofs

▶ **Lemma 21** (Theorem 4, part I). *For every $k \geq 2$, there exists a DAG, such that k `greedy` chains cover at most $(1 - 1/e)\beta_k$ vertices in the worst case, which is tight.*

Proof. We construct a class of graphs G_k for $k \in \mathbb{N}$ and show that `greedy` covers at most a $1-(1-1/k)^k$-ratio of vertices of G_k, while $\beta_k = |V|$. We introduce k^{k+1} vertices that can be covered by k paths of each k^k vertices. This set of paths represents the optimal solution and we call the i-th such path the i-th row of the graph (see Figure 2). We label the vertices accordingly: v_i^j with $i \in [k]$, $j \in [k^k]$ denotes the j-th vertex in the i-th row. The idea is that `greedy` will cover a $1/k$ fraction of the remaining uncovered vertices in each row.

The set of vertices $C_i \subseteq V(G_k)$ to be covered by the i-th `greedy` chain ($i \in [k]$) is defined as follows: $C_i = \{v_j^\ell \mid j \in [k], \ell \in \Gamma_{i,j}\}$, where the index set $\Gamma_{i,j}$ is defined as the integers inside the following interval:

$$\Gamma_{i,j} = \mathbb{N} \cap \begin{cases} [(j-i)(1-\frac{1}{k})^{i-1}k^{k-1}+1,\ (j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1}] & \text{if } j \geq i, \\ [(k+j-i)(1-\frac{1}{k})^{i-1}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1}+1, & \\ \quad (k+j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1}] & \text{if } j < i. \end{cases}$$

We add the edges to G_k that connect consecutive vertices v_j^ℓ in C_i sorted in ascending order by ℓ. Informally, the chains are paths in G_k that look like stairs, where the i-th `greedy` chain starts on the first vertex of row i and loops from row k to row 1 if $i > 1$. Clearly, G_k is a DAG, because all edges have the form $(v_{i_1}^{j_1}, v_{i_2}^{j_2})$ for some $i_1, i_2 \in [k]$ and $j_2 > j_1$. We will show the following properties:

1. The sets C_i are well-defined and pairwise disjoint,
2. The set C_i is of size $(1-\frac{1}{k})^{i-1}k^k$ and covers $(1-\frac{1}{k})^{i-1}k^{k-1}$ vertices in each row,
3. The i-th `greedy` path covers precisely the set C_i in the worst case.

Showing these three properties concludes the proof, as the number of paths chosen by `greedy` divided by the optimal solution is the following:

$$\frac{1}{\beta_k}\left|\bigcup_{i\in[k]} C_i\right| = \frac{1}{\beta_k}\sum_{i=1}^{k}|C_i| = \frac{1}{k}\sum_{i=1}^{k}\left(1-\frac{1}{k}\right)^{i-1} \xrightarrow{k\to\infty} \left(1-\frac{1}{e}\right).$$

Property 1. To verify that the sets C_i are well-defined, observe that $\Gamma_{i,j} \subseteq [k^k]$. We now verify that the sets C_i are pairwise disjoint for $k \geq 2$. Consider the sets C_i and C_{i-1} for some $i \in \{2, \ldots, k\}$ and a row $j \in [k]$. We show that the intervals in $\Gamma_{i,j}$ are disjoint by analyzing several cases:

1. If $j \geq i$, the chain of C_i is to the left of the chain of C_{i-1} (that is, the maximum value of C_i in row j is smaller than the minimum value of C_{i-1} in row j): we have $(j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1} < (j-i+1)(1-\frac{1}{k})^{i-2}k^{k-1}) + 1$, since $0 < (1-\frac{1}{k}) < 1$.
2. If $j < i-1$, the chain of C_i is also to the left of the chain of C_{i-1}: we have $(k+j-i+1)(1-\frac{1}{k})^{i-1}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1} < (k+j-i+1)(1-\frac{1}{k})^{i}k^{k-1} + (\sum_{\ell=1}^{j}(1-\frac{1}{k})^{\ell-1})k^{k-1}+1$, again, since $0 < (1-\frac{1}{k}) < 1$.
3. C_k is to the right of C_1 on all rows $j \in [k-1]$: We must verify

$$j\left(1-\frac{1}{k}\right)^{k-1}k^{k-1} + \left(\sum_{\ell=1}^{j}\left(1-\frac{1}{k}\right)^{\ell-1}\right)k^{k-1} + 1 > jk^{k-1}, \tag{1}$$

which is true if and only if

$$k^{k-1}\left[j\left(1-\frac{1}{k}\right)^{k-1} + \sum_{\ell=1}^{j}\left(1-\frac{1}{k}\right)^{\ell-1} - j\right] + 1 > 0.$$

Computing the geometric sum, we have

$$k^{k-1}\left[k\left(1-\left(1-\frac{1}{k}\right)^{j}\right)-j\left(1-\left(1-\frac{1}{k}\right)^{k-1}\right)\right]+1>0.$$

It suffices to show that the expression in the bracket is non-negative for all pairs $k > j \geq 1$. This is implied by

$$\begin{aligned}&(k-1)\left(1-\left(1-\frac{1}{k}\right)^{j}\right)\geq j\left(1-\left(1-\frac{1}{k}\right)^{k-1}\right)\\ \iff &\frac{\left(1-\left(1-\frac{1}{k}\right)^{j}\right)}{j}\geq\frac{\left(1-\left(1-\frac{1}{k}\right)^{k-1}\right)}{k-1},\end{aligned}$$

which is true, because the function $f_k(n) = (1-(1-(1/k))^n)(1/n)$ is decreasing. Hence, Equation (1) is true.

The three cases above show that C_i and C_j are disjoint for all $i \neq j \in [k]$.

Property 2. The size of C_i can be computed by summing the k interval lengths of $\Gamma_{i,j}$. Note that by Property 1, the C_i are pairwise disjoint and that the intervals $\Gamma_{i,j}$ of a path C_i all have equal length.

Property 3. To show that `greedy` takes the path C_ℓ, we need to show that C_ℓ covers the most uncovered vertices out of any path after `greedy` took $C_1, \ldots, C_{\ell-1}$. After covering the first $\ell - 1$ paths and before covering path ℓ, we say that `greedy` is at step ℓ. Given a vertex v_i^j, consider the set $\mathcal{R}_{i,j,\ell}$ of vertices that are in the same row, are uncovered at step ℓ and come after v_j^j:

$$\mathcal{R}_{i,j,\ell} = \{v_i^p \mid p > j,\ v_i^p \text{ uncovered at step } \ell\}.$$

We call an edge $(v_{i_1}^{j_1}, v_{i_2}^{j_2})$ *forward* at step ℓ if $|\mathcal{R}_{i_1,j_2,\ell}| > |\mathcal{R}_{i_2,j_2,\ell}|$. We show the following Lemma.

▶ **Lemma 25.** *Assume `greedy` took the paths $C_1, \ldots, C_t$ at steps $1, \ldots, t$. Then at step $t+1$, all edges are forward.*

Proof. We prove this fact by induction on t. All edges are forward when $t = 1$, before `greedy` chose a path. Assume $t > 0$, and note that edges that remain in the same row are obviously forward. Consider the edges $(v_{i_1}^{j_1}, v_{i_2}^{j_2})$ of a path C_ℓ whose tail is contained in a different row than its head (i.e., $j_1 \neq j_2$). If $\ell = t$, then the edges are forward: by the induction hypothesis, they are forward at step t, and after covering C_t, we have $\mathcal{R}_{i_1,j_1,t} = \mathcal{R}_{i_1,j_1,t+1}$ and $\mathcal{R}_{i_2,j_2,t} = \mathcal{R}_{i_2,j_2,t+1} \setminus C_t$.

Let now $\ell \neq t$. Following the path C_ℓ, it first is to the left of C_t and then to the right of C_t. For example, in Figure 2 the maximum value of C_2 in row 2 is smaller than the minimum value of C_3 in row 3, but it is the other way round in row 3. This is the only edge of C_ℓ that needs attention, as for every other edge $(v_{i_1}^{j_1}, v_{i_2}^{j_2})$ of C_ℓ with $j_1 \neq j_2$, we have $|\mathcal{R}_{i_1,j_1,t}| - |\mathcal{R}_{i_2,j_2,t}| = |\mathcal{R}_{i_1,j_1,t+1}| - |\mathcal{R}_{i_2,j_2,t+1}|$.

We consider the case $\ell > t$, as the other case is symmetric. The path C_ℓ is to the left of C_t on rows $\ell, \ldots, k$ and $1, \ldots, t-1$. Consider the intervals $\Gamma_{\ell,t-1}$ and $\Gamma_{\ell,t}$, and note that the latter contains an additional summand $(1-\frac{1}{k})^{t-1}k^{k-1}$, which is equal to the length of C_t in row t. Hence, this edge is also forward. ◀

Using Lemma 25, we now argue that `greedy` selects the paths C_i in the worst case. Consider the first `greedy` step, i.e., before any path was chosen by `greedy`. Every row has the same length k^k. Using Property 2, the path C_1 also has length k^k, and has the same length k^{k-1} in every row. By Lemma 25, no path is longer than the rows. Thus, in the worst case, `greedy` chooses C_1. Using induction, we keep this invariant for every path: in the i-th step, every row has the same number of uncovered vertices, the path C_i contains the same number of uncovered vertices, and no path contains more uncovered vertices. ◀