Width Helps and Hinders Splitting Flows

Manuel Cáceres
Department of Computer Science,
University of Helsinki, Finland

Massimo Cairo
Department of Computer Science,
University of Helsinki, Finland

Andreas Grigorjew
Department of Computer Science,
University of Helsinki, Finland

Shahbaz Khan
Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, India

Brendan Mumey
School of Computing, Montana State University,
Bozeman, MT, USA

Romeo Rizzi
Department of Computer Science,
University of Verona, Italy

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

Lucia Williams
School of Computing, Montana State University,
Bozeman, MT, USA

Abstract

Minimum flow decomposition (MFD) is the NP-hard problem of finding a smallest decomposition of a network flow X on directed graph G into weighted source-to-sink paths whose superposition equals X. We focus on a common formulation of the problem where the path weights must be non-negative integers and also on a new variant where these weights can be negative. We show that, for acyclic graphs, considering the *width* of the graph (the minimum number of s-t paths needed to cover all of its edges) yields advances in our understanding of its approximability. For the non-negative version, we show that a popular heuristic is a $O(\log |X|)$-approximation ($|X|$ being the total flow of X) on graphs satisfying two properties related to the width (satisfied by e.g., series-parallel graphs), and strengthen its worst-case approximation ratio from $\Omega(\sqrt{m})$ to $\Omega(m/\log m)$ for sparse graphs, where m is the number of edges in the graph. For the negative version, we give a $(\lceil \log \|X\| \rceil + 1)$-approximation ($\|X\|$ being the maximum absolute value of X on any edge) using a power-of-two approach, combined with parity fixing arguments and a decomposition of unitary flows ($\|X\| \leq 1$) into at most width paths. We also disprove a conjecture about the linear independence of minimum (non-negative) flow decompositions posed by Kloster et al. [ALENEX 2018], but show that its useful implication (polynomial-time assignments of weights to a given set of paths to decompose a flow) holds for the negative version.

2012 ACM Subject Classification Theory of computation → Network flows

Keywords and phrases Flow decomposition, approximation algorithms, graph width

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.31

Related Version *Full Version*: `https://arxiv.org/abs/2207.02136` [5]

Funding This work was partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 851093, SAFEBIO), partially by the Academy of Finland (grants No. 322595, 328877), and partially by the National Science Foundation (NSF) (grants No. 1661530,1759522).

1 Introduction

Minimum flow decomposition (MFD) is the problem of finding a smallest sized decomposition of a network flow X on directed graph $G = (V, E)$ into weighted source-to-sink paths whose superposition equals X. We focus on the case where path weights are restricted to be integers. It is a textbook result [1] that if G is acyclic (a DAG) a decomposition using no more than $m = |E|$ paths always exists. However, MFD is strongly NP-hard [25], even on DAGs, and even when the flow values come only from $\{1, 2, 4\}$ [12]. Recent work has shown that the problem is FPT in the size of the minimum decomposition [14] and that it can be formulated as an ILP of quadratic size [7].

While difficult to solve, MFD is a key step in many applications. For example, MFD on DAGs is used to reconstruct biological sequences such as RNA transcripts [18, 23, 11, 3, 22, 26] and viral strains [2]. MFD can also be used to model problems in networking [25, 12, 15] and transportation planning [16], although in some of these applications there may be cycles in the input. Despite the ubiquity of the MFD problem, the gap in our knowledge about the approximability of MFD is large, even when restricting to DAGs. It is known [12] that MFD (even on DAGs) is APX-hard (i.e., there is some $\epsilon > 0$ such that it is NP-hard to approximate within a $(1 + \epsilon)$ factor), so in particular, MFD does not admit a PTAS, unless P=NP. Furthermore, it can be approximated with a factor of $\lambda^{\log \|X\|} \log \|X\|$ [15], where λ is the length of the longest source-to-sink path and $\|X\|$ is the largest flow value in the network. In this work, we attempt to fill in some of the gaps between these results.

A natural lower bound for the size of an MFD of a DAG is the size of a *minimum path cover* of the set of edges with non-zero flow (i.e., the minimum number of paths such that every such edge appears in *at least* one path) – this size is called the *width* of the network. This trivially holds because every flow decomposition is also such a path cover. These two notions are analogies of the more standard notions of path cover and width of the *node set*. The node-variants are classical concepts, with algorithmic results dating back to Dilworth and Fulkerson [8, 10]. Despite this, the width has not been given any attention in the MFD problem, and in particular it has never been used in approximation algorithms. In this paper, we show that the width can play a key role both in the analysis of popular heuristics, and in obtaining the first approximation algorithm for a natural variant of MFD.

We start with a relaxation of MFD in which flow decomposition may also use negative integer weighs on flow paths, rather than strictly positive weights as has traditionally been considered [25, 12, 14]. An important observation that we leverage for this variant (unlike the positive-only version) is that "the width does not increase" as flow paths are chosen and removed. Using this, we give a $(\lceil \log \|X\| \rceil + 1)$ approximation algorithm for this variant. To differentiate both versions, we use $\text{MFD}_{\mathbb{N}}$ and $\text{MFD}_{\mathbb{Z}}$ throughout the paper. While $\text{MFD}_{\mathbb{Z}}$ is a natural version of the problem, to our knowledge it has not been previously considered in the MFD literature. However, it can also have natural applications, since by applying $\text{MFD}_{\mathbb{Z}}$ on the difference between two flows, one can minimally explain the *differences* between them, e.g. to explain the differences in RNA expression between two tissue samples with the fewest number of up/down regulated transcripts, which is often the goal of RNA sequencing experiments [21]. Our approximation follows a *power-of-two* approach where the weights of the paths chosen are (positive or negative) powers of two. More specifically, observe that if all flow values are even, then one can divide them by 2 and obtain a flow X with smaller $\|X\|$ whose decomposition can be transformed back into a decomposition of X. In order to obtain such an even flow, we prove a basic property that can be of independent interest: given any integer flow X, there exists a *unitary* flow (its values are 0, +1, or −1) Y, such that $X + Y$ is even on every edge (Lemma 5). In addition, given a unitary flow Y, we show that Y can be decomposed into k paths of weight +1 or −1, such that k is at most the width of the graph (Corollary 8). We obtain the $(\lceil \log \|X\| \rceil + 1)$ approximation ratio (Theorem 11) by iteratively removing the unitary flow, dividing all flow values by 2, and preprocessing the graph so that its width is a lower bound on the size of the $\text{MFD}_{\mathbb{Z}}$.

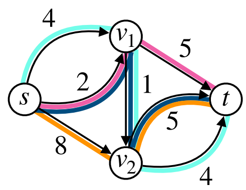
In Section 4 we consider connections between the width and a popular heuristic algorithm for $\mathrm{MFD}_{\mathbb{N}}$ which we call *greedy-weight*[1] [25], which builds a flow decomposition by successively choosing the path that can carry the largest flow. Greedy-weight is commonly used in applications (see e.g., [23, 2, 18] among many), and it seems to be mentioned in nearly every publication addressing flow decomposition. First, on sparse graphs we improve (i.e., increase) the worst-case lower bound for the greedy-weight approximation factor from $\Omega(\sqrt{m})$ [12] to $\Omega(m/\log m)$, showing for the first time that greedy-weight can be exponentially worse than the optimum. For this we use a class of sparse graphs where the optimum flow decomposition has size $O(\log m)$ whereas the greedy-weight algorithm returns a solution of size $\Omega(m)$, only a constant factor away from the trivial decomposition. The key to this new bound is to design an input where the width increases exponentially when a path is greedily removed. We also show that the same bound also holds for other greedy heuristics choosing instead the longest or shortest paths. Second, we show that if the input satisfies the properties that its width does not increase as source-to-sink paths are removed (Property 15) and that it is possible to remove a path of large weight (Property 16), then greedy-weight is a $O(\log |X|)$-approximation, where $|X|$ is the flow value (i.e., total flow out of s). A notable class of graphs satisfying these properties is the class of *series-parallel graphs*; see [9, 24] for fast recognition algorithms and pointers to other NP-hard problems that are easier on this class of graphs. Series-parallel graphs are also of great interest for network flow problems (see, e.g., [13, 4]).

Finally, in Section 5 we consider a closely related problem, called *k-Flow Weight Assignment* [14]. In addition to the flow X, in this problem we are also given a set of k paths, and we need to decide if there is an assignment of weights to the paths such that they form a decomposition of X. If the weights belong to $\mathbb{N}$, this was shown to be NP-complete in [14]. In this work, we first observe that in the same way that allowing negative integer weights simplifies the approximability of MFD, allowing weights to belong to $\mathbb{Z}$ fully changes the complexity of the k-Flow Assignment Problem, making it polynomial. This is due to the fact that the linear system defined by the given paths loses its only inequality of restricting the weights to positive integers. It thus transforms an ILP to a system of linear Diophantine equations, which can be solved in polynomial time (see e.g. [19]). Second, we consider a conjecture from [14] stating that if the weights belong to $\mathbb{N}$, and k is the size of a $\mathrm{MFD}_{\mathbb{N}}$ for X, then the problem admits a unique solution (i.e., a unique assignment of weights to the given paths). If true, this would speed up the FPT algorithm of [14] for $\mathrm{MFD}_{\mathbb{N}}$, because a step solving an ILP could be executed by solving a standard linear program returning a rational solution and checking if the (supposedly unique) solution to this system is integer. Moreover, the same conjecture (with the same implication) was also a motivation behind the greedy algorithm of [20] for $\mathrm{MFD}_{\mathbb{N}}$. In this paper, we disprove the conjecture of [14], further corroborating the gap between $\mathrm{MFD}_{\mathbb{N}}$ and $\mathrm{MFD}_{\mathbb{Z}}$.

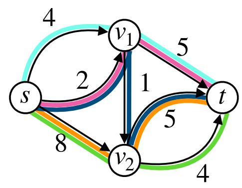
2 Preliminaries

We are given a directed graph $G = (V, E)$. Without loss of generality, we assume a unique source s and a unique sink t with no in-edges and no out-edges respectively; otherwise, the graph can be converted to such a graph by adding a pseudo source and sink and connecting them to all sources and sinks respectively with appropriately weighted edges. We also assume

[1] Previous work has consistently referred to this algorithm as *greedy-width*. To avoid confusion with the width of the graph, we introduce the name greedy-weight in this work.



(a) If negative weights are allowed, the four paths decompose the flow with weights $4, 5, 8$, and -3 (dark blue).



(b) With positive weights only, five paths are needed, since the edge (v_1, v_2) must be decomposed by a weight 1 path, leaving 4 edges that must be covered separately. The paths shown are one such decomposition.

Figure 1 A positive flow admitting a decomposition into four paths only if negative weights are allowed.

that every node is on some s-t path. We use n and m to denote the number of nodes and edges of G, respectively. Additionally, we assume that G is a DAG throughout the paper. While the problem is also studied for graphs with cycles (see, e.g., [25, 12]), the task is still to decompose into simple paths, and so our inapproximability result on DAGs also applies for graphs with cycles. We call functions $X : E \to \mathbb{Y}$ *pseudo-flows*, where $\mathbb{Y}$ is some set of allowed flow values (numbers). We treat pseudo-flows as vectors over E and use the notation $X + Y$ and aX to denote the (element-wise) sum of pseudo-flows and multiplication by a scalar, respectively. The numbers 0 and 1 also denote (depending on context) pseudo-flows that are 0 (resp. 1) everywhere. We write $X \leq Y$ (and similarly $<$) to mean $X(u, v) \leq Y(u, v)$ for every $(u, v) \in E$.

Given G, a *flow* is a pseudo-flow satisfying conservation of flow (incoming flow equal to outgoing flow) on internal nodes $V \setminus \{s, t\}$. It is known that the sum of two flows $X + Y$, the multiplication of a flow with a scalar aX, and the empty flow 0 are themselves flows. Let $|X|$ denote the total flow out of s (or into t) for flow X. Given an s-t path P, let P also denote the flow defined by setting 1 to every edge in P and 0 elsewhere. With these definitions, we are ready to formally define MFD.

▶ **Definition 1.** *Given a flow X, a* flow decomposition *of (G, X) of* size k *is a family of s-t paths $\mathcal{P} = (P_1, \dots, P_k)$ with weights $(w_1, \dots, w_k) \in \mathbb{Y}^k$ such that $X = w_1 P_1 + \dots + w_k P_k$.*

▶ **Definition 2.** *Given a flow X, let $\mathsf{mfd}_{\mathbb{Y}}(G, X)$ be the smallest size of a flow decomposition of (G, X) with weights in $\mathbb{Y}$.*

We omit $\mathbb{Y}$ if it is clear from the context. We call the problem of producing a flow decomposition of (G, X) of minimum size the *minimum flow decomposition (MFD) problem*. In this paper, we study two integer versions of the problem, $\text{MFD}_{\mathbb{N}}$ ($0 \in \mathbb{N}$) and $\text{MFD}_{\mathbb{Z}}$. Note that the reduction showing $\text{MFD}_{\mathbb{N}}$ to be strongly NP-hard from [25] also holds for $\text{MFD}_{\mathbb{Z}}$. However, a positive flow may admit a decomposition using fewer paths if negative weights are allowed, as shown in Figure 1. We explore further differences between $\text{MFD}_{\mathbb{N}}$ and $\text{MFD}_{\mathbb{Z}}$ in Sections 3 and 5.

Let $\|X\| = \max_{(u,v) \in E} |X(u, v)|$ denote the infinity norm on flows. In particular, notice that if $\mathbb{Y} \subseteq \mathbb{Z}$, then $\|X\| \leq 1$ means that $X(u, v) \in \{0, \pm 1\}$ for every $(u, v) \in E$. Let $X \equiv_2 Y$ if X and Y have the same parity everywhere, i.e., for every $(u, v) \in E$, we have that $X(u, v)$ is odd iff $Y(u, v)$ is odd.

▶ **Definition 3.** *Given $S \subseteq E$, we define $\mathsf{width}_S(G)$ as the minimum number of s-t paths in G needed to cover all edges of S. If $S = E$ we just write $\mathsf{width}(G)$.*

Just like its more common node variant, $\mathsf{width}(G)$ can be computed in $O(mn)$ time. As described by, e.g., [1, 6], this is done by reduction to a min-flow instance with demand one on every edge; the minimum flow of this instance is $\mathsf{width}(G)$, and the flow can be found by reduction to a max-flow instance. Moreover, the problem can be relaxed to only require the coverage of $S \subseteq E$ and solved in the same running time by setting the demands only on the edges of S.

▶ **Lemma 4** ([1, 17]). *Let $G = (V, E)$ be a DAG, and $S \subseteq E$. A flow $C : E \to \mathbb{N}$ can be computed in $O(mn)$ time, such that $C(e) \geq 1$ for all $e \in S$ and $|C| = \mathsf{width}_S(G)$.*

The flow C with total flow $\mathsf{width}_S(G)$ suffices in this paper, and we do not need to calculate a path cover achieving that minimum. However, we note that it can be directly computed given the flow C. We can think of this path cover as a flow decomposition of C into $\mathsf{width}_S(G)$ weight-one paths, which can be found by greedily removing such paths from C until it is completely decomposed. Since each path has no more than $n - 1$ edges and since $\mathsf{width}_S(G) \leq m$, the overall runtime of finding the path cover is $O(mn)$.

3 Width helps solve $\mathrm{MFD}_{\mathbb{Z}}$

The idea behind our approximation algorithm for $\mathrm{MFD}_{\mathbb{Z}}$ is that a flow $X : E \to \mathbb{Z}$ on DAG G can always be decomposed into $(\lceil \log \|X\| \rceil + 1) \cdot \mathsf{width}(G)$ paths. We show this using two key facts: first, that X can be decomposed into $(\lceil \log \|X\| \rceil + 1)$ flows with a particular structure, and, second, that each of these flows can be decomposed into $\mathsf{width}(G)$ paths. A key step in proving both these facts is a subroutine which, given an input flow X, finds another flow Y with only values from $\{0, \pm 1\}$ (a *unitary* flow) that matches the parity of X on all the edges. Intuitively, given an input flow X, such a unitary flow Y can be added to X to "fix" its odd edges to be even, with only a small change to X.

▶ **Lemma 5.** *For any flow $X : E \to \mathbb{Z}$ on $G = (V, E)$, there exists a flow $Y : E \to \mathbb{Z}$ such that $X \equiv_2 Y$ and $\|Y\| \leq 1$.*

Proof. Consider the undirected graph $G_{\text{odd}} = (V, E_{\text{odd}})$ where $E_{\text{odd}} = \{\{u, v\} \mid (u, v) \in E \text{ and } X(u, v) \text{ is odd}\}$.

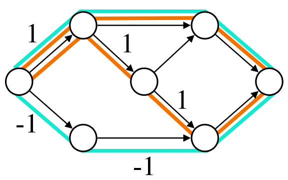
Notice that every node of G_{odd}, except possibly s and t, has even degree, due to conservation of flow. Thus, G_{odd} can be written as the edge-disjoint union of cycles and s-t paths. Assign an arbitrary orientation to each cycle and s-t path, and let E^+_{odd} be the set of edges oriented in this way. Define

$$Y(u, v) = \begin{cases} +1 & \text{if } (u, v) \in E^+_{\text{odd}} \\ -1 & \text{if } (v, u) \in E^+_{\text{odd}} \\ 0 & \text{if } \{u, v\} \notin E_{\text{odd}} \end{cases}$$

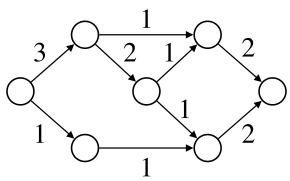
Notice that Y is a flow decomposed as a sum of flows, each along one of the edge-disjoint cycles and s-t paths. Moreover, $X \equiv_2 Y$ and $\|Y\| \leq 1$ by construction. ◀

Repeatedly applying Lemma 5 and dividing the resulting even flow by 2, we obtain the the first key ingredient of the approach (proof in [5, Appendix B]).

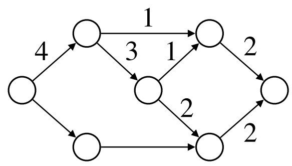
▶ **Corollary 6.** *Any (non-zero) flow $X : E \to \mathbb{Z}$ can be written as $X = \sum_{i=0}^{\lceil \log \|X\| \rceil} 2^i \cdot Y_i$, where $Y_i : E \to \mathbb{Z}$ is a flow with $\|Y_i\| \leq 1$ for all i.*



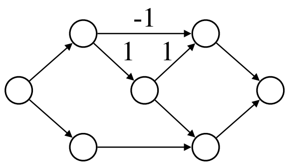
(a) Unitary flow X on a graph G and a decomposition of it into four paths, two of weight 1 in orange (see **(e)**), and two of weight -1 in blue (see **(f)**).



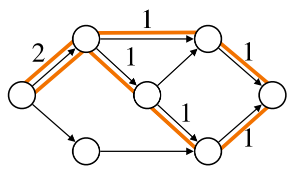
(b) Flow C covering all edges of G, of size $|C| = \mathsf{width}(G) = 4$ (Lemma 4).



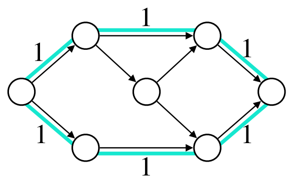
(c) Flow $X + C$.



(d) Unitary flow D matching the parity of $X{+}C$, i.e., $D \equiv_2 X{+}C$ (Lemma 5).



(e) Flow $A = (C - D + X)/2$ and a decomposition of it into two paths of weight 1.



(f) Flow $B = (C - D - X)/2$ and a decomposition of it into two paths of weight 1.

Figure 2 Example of Lemma 7 and Corollary 8 applied to a unitary flow X on a graph G (for clarity, 0 flow values are not shown). Positive flows A and B can be constructed so that $|A| + |B| \leq \mathsf{width}(G)$ holds. Flows A and B can be trivially decomposed into $|A|$ and $|B|$ paths, respectively. We obtain a decomposition of X by taking the paths of A with weight 1 and the paths of B with weight -1.

The following result is the second key ingredient of our approach. It guarantees (together with Corollary 8) that any unitary flow can be decomposed into at most $\mathsf{width}(G)$ paths of weight ± 1 (see Figure 2 for an example). This is by no means obvious since, among other problems, a unitary flow may contain positive and negative values which merge and cancel each other out (as in Figure 2a). The proof is based on another application of Lemma 5, along with some algebra on flows.

▶ **Lemma 7.** *For any flow $X : E \to \mathbb{Z}$, $\|X\| \leq 1$, there exist flows $A, B : E \to \mathbb{Z}$ such that:*
1. $A, B \geq 0$
2. $X = A - B$
3. $|A| + |B| \leq \mathsf{width}(G)$

Proof. Take C such that $C \geq 1$ and $|C| = \mathsf{width}(G)$, according to Lemma 4. Take D such that $D \equiv_2 X + C$ and $\|D\| \leq 1$, according to Lemma 5. Also, assume $|D| \geq 0$ without loss of generality (otherwise, take $-D$, which satisfies the same properties).

Since $D \equiv_2 X + C$, we have $C - D \pm X \equiv_2 0$. So we can take $A = (C - D + X)/2$ and $B = (C - D - X)/2$.

1. Notice that $C - D \pm X \geq C - 2$ since $\|D\|, \|X\| \leq 1$. So, $C - D \pm X \geq -1$, since $C \geq 1$. But $C - D \pm X \equiv_2 0$ so $C - D \pm X \geq 0$, whence $A, B \geq 0$.
2. $A - B = \frac{C-D+X}{2} - \frac{C-D-X}{2} = X$.
3. $|A| + |B| = |A + B| = \left|\frac{C-D+X}{2} + \frac{C-D-X}{2}\right| = |C - D| = |C| - |D| \leq |C|$ since $|D| \geq 0$, and $|C| = \mathsf{width}(G)$. ◀

▶ **Corollary 8.** *For any flow $X : E \to \mathbb{Z}$ with $\|X\| \leq 1$, there exist paths $P_1, \ldots, P_k$ with $k \leq \mathsf{width}(G)$ such that $X = P_1 + \cdots + P_\ell - P_{\ell+1} - \cdots - P_k$ (for some $0 \leq \ell \leq k$).*

Proof. Take A, B according to Lemma 7, with $A, B \geq 0$, $X = A - B$ and $|A| + |B| \leq \mathsf{width}(G)$. Since $A, B \geq 0$, there exist paths $P_1, \ldots, P_{|A|+|B|}$ such that $A = P_1 + \cdots + P_{|A|}$ and $B = P_{|A|+1} + \cdots + P_{|A|+|B|}$ [1]. Since $X = A - B$, we can write $X = P_1 + \cdots + P_{|A|} - P_{|A|+1} - \cdots - P_{|A|+|B|}$. Finally, recall that $|A| + |B| \leq \mathsf{width}(G)$, concluding the proof. ◀

Finally, expressing any flow as a sum of at most $\lceil \log \|X\| \rceil + 1$ unitary flows (Corollary 6), and decomposing each unitary flow into at most $\mathsf{width}(G)$ positive or negative paths (Corollary 8), we can decompose the flow into at most $\lceil \log \|X\| \rceil + 1$ paths whose weight are positive and negative powers of two.

▶ **Theorem 9.** *Given a DAG $G = (V, E)$, for any flow $X \colon E \to \mathbb{Z}$, there exist paths $P_1, \ldots, P_k$ and weights $\{w_1, \ldots, w_k\} \subseteq \{\pm 2^i \mid i \in \mathbb{N}\}$, with $k \leq (\lceil \log \|X\| \rceil + 1) \cdot \mathsf{width}(G)$ such that $X = w_1 P_1 + \cdots + w_k P_k$.*

Proof. Combine Corollaries 6 and 8, getting

$$X = \sum_{i=0}^{\lceil \log \|X\| \rceil} 2^i \cdot Y_i = \sum_{i=0}^{\lceil \log \|X\| \rceil} 2^i \cdot (P_{1_i} + \cdots + P_{\ell_i} - P_{\ell_i + 1} - \cdots - P_{k_i})$$

where $k_i \leq \mathsf{width}(G)$. ◀

The proof of Theorem 9 suggests a straightforward algorithm for MFD$_{\mathbb{Z}}$, which we detail in Algorithm 2 and describe at a high level here. First, iteratively decompose X, yielding $\log\lceil \|X\| \rceil + 1$ unitary flows. Then use Lemma 7 to decompose each into $\mathsf{width}(G)$ paths. However, as we explained at the beginning of this section, $\mathsf{width}(G)$ is not a lower bound on MFD$_{\mathbb{Z}}$, and thus this approach does not directly derives an approximation. To overcome this issue, we instead find a flow decomposition of a spanning subgraph G' of G whose width lower bounds $\mathsf{mfd}_{\mathbb{Z}}(G, X)$. Namely, we first find a minimum path cover flow in G of the subset S of edges with non-zero flow in $O(mn)$ time (according to Lemma 4), and then remove from G any edge not covered by the flow, obtaining G'. By construction, the size of this path cover is a lower bound of $\mathsf{mfd}_{\mathbb{Z}}(G, X)$. Moreover, the size of this path cover is exactly $\mathsf{width}(G')$, since every path cover of G' is also a path cover of S in G.

To prove the correctness of Algorithm 2, we first define a a subroutine implementing Lemma 5.

▶ **Lemma 10.** *Algorithm 1 returns a unitary flow from input flow Y such that $X \equiv_2 Y$, as in Lemma 5, in $O(m)$ time.*

Proof. The correctness of the algorithm is given by Lemma 5. Finally, the first 3 subroutines as well as the entire `for-loop` takes $O(m)$ time. ◀

▶ **Theorem 11.** *Algorithm 2 is a $\log\lceil \|X\| \rceil + 1$-approximation for MFD$_{\mathbb{Z}}$ with runtime $O(m(\log \|X\| \cdot \mathsf{mfd}_{\mathbb{Z}}(G, X) + n)) = O(m^2 \log \|X\|)$.*

Proof. By Theorem 9 and our previous discussion, Algorithm 2 returns a flow decomposition for X with no more than $(\lceil \log \|X\| \rceil) + 1 \cdot \mathsf{width}(G') = (\lceil \log \|X\| \rceil + 1) \cdot \mathsf{mfd}_{\mathbb{Z}}(G, X)$ paths. We analyze the runtime line by line. Lines 2 and 5 take $O(mn)$ time by Lemma 4. The call to Algorithm 1 on line 6 takes $O(m)$ time by Lemma 10, and checking the flow of D and flipping signs (if necessary) also takes $O(m)$ time. By Corollary 6, the while loop on line 8 executes at most $\log\lceil \|X\| \rceil + 1$ times, meaning that the entire execution takes $O(m \log \|X\|)$ time since line 9 takes $O(m)$ time by Lemma 10. Since there are at most $\log\lceil \|X\| \rceil + 1$ Y_i's, the for loop on line 14 executes at most $\log\lceil \|X\| \rceil + 1$ times. Each

Algorithm 1 Produces a unitary flow Y from input flow X such that $X \equiv_2 Y$, as in Lemma 5.

procedure $\textsc{Unitary}(G, X)$
    $E_{\text{odd}} \leftarrow$ odd edges of G, undirected
    $C \leftarrow$ a decomposition of $G_{\text{odd}} = (V, E_{\text{odd}})$ into cycles, oriented arbitrarily
    $E^+_{\text{odd}} \leftarrow$ directed edges of C
    for $(u, v) \in E$ **do**
        if $(u, v) \in E^+_{\text{odd}}$ **then**
            $Y(u, v) \leftarrow +1$
        else if $(v, u) \in E^+_{\text{odd}}$ **then**
            $Y(u, v) \leftarrow -1$
        else
            $Y(u, v) \leftarrow 0$
        end if
    end for
    return Y
end procedure

execution of the for-loop naively finds $\mathsf{width}(G')$ paths, each of which can be found in $O(m)$ time, so the whole loop takes $O(\log \|X\| \cdot \mathsf{width}(G') \cdot m)$ time. Thus, the overall runtime is $O(m \log \|X\| \cdot \mathsf{width}(G') + n) = O(m \log \|X\| \cdot \mathsf{mfd}_{\mathbb{Z}}(G, X) + n)$. ◀

A theorem analogous to Theorem 9 for $\text{MFD}_{\mathbb{N}}$ is desirable, but cannot be achieved directly with the previous methods. Lemma 5 makes use of negative weights, and yields positive weights only if the flow graph solely consists of s-t paths. However, the approach can be adapted for $\text{MFD}_{\mathbb{N}}$ if the input flows are stable (Property 15), and if it is possible to "fix" the odd flows to be even with only $\mathsf{width}(G)$ paths, which we leave as an open question.

4 Width matters for greedy approaches

Since the difference of two flows is still a flow, it is very natural to consider successively removing the most obvious type of flow – that is to say, paths – as an algorithmic strategy for $\text{MFD}_{\mathbb{N}}$. Indeed, the particular greedy path removal strategy of finding the *heaviest path* (*greedy-weight*) is commonly used as a heuristic in applications (e.g., [18, 2, 23, 12]) and it seems to be mentioned in nearly every paper addressing flow decomposition. More formally, a path P is said to *carry* flow p if $X(u, v) \geq p$ for all edges (u, v) of P. The *heaviest path* is an s-t path carrying the largest flow. Such a path can be easily found in linear time in the size of the DAG by dynamic programming (see, e.g., [25]).

4.1 Width hinders greedy on $\text{MFD}_{\mathbb{N}}$

We define a family of $\text{MFD}_{\mathbb{N}}$ instances $(G_\ell, X_{\ell,B})$, depending on two parameters $\ell \in \mathbb{N} \setminus \{0\}$ and $B \in \mathbb{N}$. The family is defined recursively on ℓ. The base case $(G_1, X_{1,B})$ for $\ell = 1$ is shown in Figure 3a. For $\ell > 1$, we build $(G_\ell, X_{\ell,B})$ from two disjoint copies of $(G_{\ell-1}, X_{\ell-1,B})$, by adding 5 extra edges and flow values as shown in Figure 3b. We call the edge connecting the two copies of $G_{\ell-1}$ a *central edge*. Edges whose flow value depends on B are called *backbone* edges, and they form a s-t path. Choosing $B = 2^{\ell+1}$, we show that the flow $X_{\ell,2^{\ell+1}}$ can be decomposed using a number of paths linear in ℓ, thanks to the heavy backbone

Algorithm 2 Finds the flow decomposition of Theorem 9.

procedure Path-Decomposition(G, X)
  Compute a minimum path cover flow of $\{(u,v) \in E \mid X(u,v) \neq 0\}$ ▷ Lemma 4
  Remove from G any edge not covered by this path cover to obtain G'
  $\mathcal{P} \leftarrow [], w \leftarrow []$ ▷ length-zero vectors
  $C \leftarrow$ flow of value $\mathsf{width}(G')$, $C \geq 1$ ▷ Lemma 4
  $D \leftarrow$ Unitary(G', C); if $|D| < 0$ set $D = -D$ ▷ Algorithm 1
  $i \leftarrow 0$
  while $\|X\| > 1$ **do**
    $Y_i \leftarrow$ Unitary(G', X) ▷ Algorithm 1
    $X \leftarrow (X - Y_i)/2$
    $i \leftarrow i + 1$
  end while
  $Y_i \leftarrow X$
  for $j \in \{0, \ldots, i\}$ s.t. $Y_j \neq 0$ **do**
    $A \leftarrow C - D + Y_j$, $B \leftarrow C - D - Y_j$
    Naively decompose A into $|A|$ paths and B into $|B|$ paths; concatenate A, B to $\mathcal{P}$
    Concatenate $|A|$ copies of 2^j and $|B|$ copies of -2^j to w
  end for
  return $(\mathcal{P}, w)$
end procedure

edges, whereas the greedy-weight algorithm fully saturates the central edges with its first path and is left with a remaining flow requiring $2^{\ell+1}$ paths to be decomposed (proofs in [5, Appendix B]).

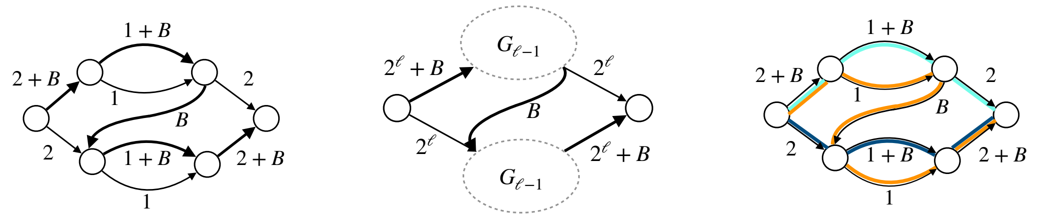
▶ **Lemma 12.** *Let G_ℓ with flow $X_{\ell,2^{\ell+1}}$ be constructed as described before. Greedy-weight uses $1 + 2^{\ell+1}$ paths to decompose $X_{\ell,2^{\ell+1}}$.*

▶ **Lemma 13.** *Let G_ℓ with flow $X_{\ell,2^{\ell+1}}$ be constructed as described before. It is possible to decompose $X_{\ell,2^{\ell+1}}$ using $2\ell + 2$ paths.*

▶ **Theorem 14.** *The approximation ratio for greedy-weight on MFD$_{\mathbb{N}}$ is $\Omega(m/\log m)$ for sparse graphs, in the worst case.*

Proof. By Lemmas 12 and 13, greedy-weight uses $\Theta(2^\ell)$ paths to decompose the flow $X_{\ell,2^{\ell+1}}$ described above, whereas it is possible to decompose the flow with only $\Theta(\ell)$ paths. It can be easily verified by induction that the number of edges of G_ℓ is $7 \cdot 2^\ell - 5$. So the ratio between greedy-weight and optimal for this instance is $\Omega(\frac{m}{\log m})$. ◀

While greedy-weight is most commonly used in applications, the approach was first presented as part of a general framework [25]: pick any optimality criteria for s-t paths that is *saturating* (i.e., fully decomposes at least one edge), and successively remove optimal paths. Since each path is saturating, the algorithm must decompose the flow in m or fewer paths. Another optimality criterion sometimes used in DNA assembly (e.g., in vg-flow [2]) is the *longest path* (with its maximum possible flow so that it is saturating). To adapt our construction of $(G_\ell, X_{\ell,2^{\ell+1}})$ so that this approach has the same approximation ratio, consider $(G^*_\ell, X^*_{\ell,2^{\ell+1}})$, constructed as in $(G_\ell, X_{\ell,2^{\ell+1}})$ except that we replace every backbone edge (u,v) with two edges, (u,w) and (w,v). See [5, Figure 6] for an example. Then a path



(a) The base case $(G_1, X_{1,B})$ $(\ell = 1)$. Backbone edges (bold) carry flow at least B.

(b) Building $(G_\ell, X_{\ell,B})$ from two copies of $(G_{\ell-1}, X_{\ell-1,B})$ $(\ell > 1)$. The 5 new edges connect the source and sink of $(G_\ell, X_{\ell,B})$ with the sources and sinks of $(G_{\ell-1}, X_{\ell-1,B})$. The central edge has flow B and is part of the backbone (bold edges).

(c) Decomposing the base case $(G_1, X_{1,B})$ $(\ell = 1)$, for $B = 2^{\ell+1}$. All non-backbone edges can be decomposed with $2\ell+1 = 3$ paths. The orange path has weight 1 and dark and light blue paths have weight 2. A fourth path (of weight 3) along the backbone is required to fully decompose the flow.

Figure 3 Construction for $(G_\ell, X_{\ell,B})$. Setting $B = 2^{\ell+1}$ gives MFD instances where greedy-weight uses $\Theta(\frac{m}{\log m})$ times more paths than optimal to decompose the flow.

along the backbone edges will be the longest from s to t and the above asymptotic analysis holds, since we no more than doubled the number of edges. Yet another optimality criterion, studied in [12] for its application to network routing, is the shortest path (again with its maximum possible flow). $(G^*_\ell,\ X^*_{\ell,2^{\ell+1}})$ will also force this approach to take an exponential number of paths, since first the algorithm will take all $2^{\ell+1}$ weight-1 edges with $2^{\ell+1}$ different paths.

4.2 Greedy approximation for series-parallel graphs

As exploited in Section 4.1, one sticking point for greedy path removal algorithms is the fact that the width of a graph can increase after an edge is fully decomposed. We now show that if, in a particular instance, a graph does not increase its width during the execution of the algorithm, and greedy-weight can decompose "enough" flow at each step, then greedy-weight is a $O(\log |X|)$-approximation for $\text{MFD}_\mathbb{N}$.

If G is a directed graph and $X \geq 0$ a flow on G, we write $G|_X$ (*G restricted to X*) to mean the spanning subgraph of G made up of the edges $e \in E$ such that $X(e) \neq 0$. Conversely, if S is a subgraph of G, we write $X|_S$ (*X restricted to S*) to mean the pseudo-flow X only on the edges of S. In the case of $\text{MFD}_\mathbb{N}$, once an edge is fully decomposed, it cannot be used in future paths, possibly increasing the width of the graph that can be used to decompose the rest of the flow and sometimes triggering an increase of the size of a minimum flow decomposition as well. We call a graph *stable* if it does not have this issue.

▶ **Property 15** (Stable graph)**.** We say that G is *stable* if, for any non-negative flow X on G, it holds that $\mathsf{width}(G|_X) \leq \mathsf{width}(G)$.

Many useful $\text{MFD}_\mathbb{N}$ instances do in fact satisfy Property 15. For example, the first proof of MFD's NP-hardness [25] was a reduction to a very simple graph of this form; this means that $\text{MFD}_\mathbb{N}$ restricted to stable graphs is also NP-hard.

The second property that we need is that there is always, during the execution of the algorithm, a path carrying "enough" flow from s to t.

▶ **Property 16** (Paths of large weight)**.** We say that G *has paths of large weight* if, for any flow $X \geq 0$ on G, there exists an s-t path in $G|_X$ carrying at least $|X|/\mathsf{width}(G)$ flow.

Note that this property does not hold in general; see [5, Figure 7].

▶ **Lemma 17.** *Let $G = (V, E)$ be a graph, $\mathsf{width}(G) \geq 2$, satisfying Properties 15 and 16. Greedy-weight uses at most $\lfloor \log |X| / \log \frac{\mathsf{width}(G)}{\mathsf{width}(G)-1} \rfloor + 1$ paths to decompose any flow $X : E \to \mathbb{N}$.*

Proof. Let $b = \mathsf{width}(G)$. Since G satisfies Properties 15 and 16, greedy-weight removes a path of weight at least $|X'|/b$ at every step, where X' is the remaining flow of the corresponding step. As such, after c steps $|X'| \leq |X| \left(\frac{b-1}{b}\right)^c$. If $|X| \left(\frac{b-1}{b}\right)^c < 1$, then $|X'| = 0$, since $|X|$ and the weights of the removed paths belong to $\mathbb{N}$. Solving for c we obtain $c > \log |X| / \log \frac{b}{b-1}$. Therefore, greedy-weight takes (uses) at most $c = \lfloor \log |X| / \log \frac{b}{b-1} \rfloor + 1$ steps (paths). ◀

▶ **Theorem 18.** *Let $G = (V, E)$ be a graph satisfying Properties 15 and 16 and $X : E \to \mathbb{N}$ a flow. Greedy-weight is a $O(\log |X|)$-approximation for MFD$_\mathbb{N}$ on (G, X).*

Proof. Assume $X > 0$ (otherwise, replace G with $G|_X$). Thus, $b = \mathsf{width}(G) \leq \mathsf{mfd}_\mathbb{N}(G, X)$, since any flow-decomposition of X induces a path cover of E. If $b \leq 1$ greedy-weight finds an optimal solution. Otherwise $b \geq 2$, and Lemma 17 implies that greedy-weight is a $O(\frac{\log |X|}{b \log \frac{b}{b-1}})$ $=O(\log |X|)$-approximation for MFD$_\mathbb{N}$ ($b \log \frac{b}{b-1} = O(1)$ for $b \geq 2$). ◀

Finally, we define series-parallel graphs, and apply Theorem 18 to them, by proving (in [5, Appendix B]) that they satisfy Properties 15 and 16.

▶ **Definition 19** (Series-parallel graph [9])**.** *A graph is a* two-terminal series-parallel *(*series-parallel *for short) graph with terminal nodes s and t if:*

- *it consists of a single edge directed from s to t, and no other nodes,* or
- *it can be obtained from two (smaller) two-terminal series-parallel graphs G_1 and G_2, with terminal nodes s_1, t_1, and s_2, t_2, respectively, by either*
  - *identifying $s = s_1 = s_2$ and $t = t_1 = t_2$ (*parallel composition *of G_1 and G_2), or*
  - *identifying $s = s_1$, $t_1 = s_2$, and $t = t_2$ (*series composition *of G_1 and G_2).*

▶ **Corollary 20.** *Greedy-weight is a $O(\log |X|)$-approximation for MFD$_\mathbb{N}$ on series-parallel graphs.*

5 Solving the k-Flow Weight Assignment Problem

In this section, we consider a restriction of MFD from [14] (see Figure 4 for an example).

▶ **Definition 21** (k-Flow Weight Assignment)**.** *Given a flow $X : E \to \mathbb{Y}$ on a graph $G = (V, E)$ and a set of s-t paths $\{P_1, \ldots, P_k\}$, the problem of finding an assignment of weights to the paths, such that they form a flow decomposition of (G, X), is called k-*Flow Weight Assignment *(k-FWA). We write k-FWA$_\mathbb{Y}$ if we require the path weights to belong to $\mathbb{Y}$.*

Given k s-t paths, k-FWA can be solved by a linear system defined by $Lw = X$, where $X_j \in \mathbb{Y}$ is equal to the flow $X(e_j)$ of the edge e_j (we identify flows $X : E \to \mathbb{Y}$ with vectors $X \in \mathbb{Y}^m$) and L is the $m \times k$ 0/1 matrix with $L_{i,j} = 1$ if and only if path P_j crosses edge e_i. The resulting solution $w \in \mathbb{Y}^k$ is the weight assignment to each path. For a flow graph (G, X), we denote by $L_\mathbb{Y} = L_\mathbb{Y}(P_1, \ldots, P_k) = \{w \in \mathbb{Y}^k | X = \sum_{j=1}^k P_k w_k\}$ the linear system corresponding to the paths $P_1, \ldots, P_k$.

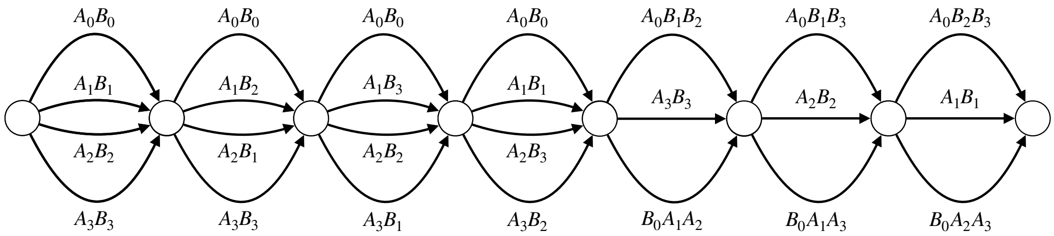


Figure 4 Paths A_i and B_i ($i \in \{0, 1, 2, 3\}$), each edge being labeled with the paths it appears in. Assign to each path A_i weight a_i, and to each path B_i weight b_i, such that $a_0 = b_0 = 3$, and $a_i = 6^{2i}+1$ and $b_i = 6^{2i+1} + 5$ for $i = 1, 2, 3$. Define the flow X on G as $X = \sum_{i=0}^{3} a_iA_i + \sum_{i=0}^{3} b_iB_i$. Note that these weights are a solution of k-FWA$_{\mathbb{N}}$ on input (G, X) with given paths A_i, B_i ($i \in \{0, 1, 2, 3\}$).

We shortly discuss how to solve k-FWA$_{\mathbb{Z}}$. It is possible to find an integer-weighted solution or decide that it does not exist in polynomial time. The reason for this is that, having no non-negativity constraint, the linear system defined by the paths is a system of linear Diophantine equations. It is well known that integer solutions to such systems can be found in polynomial time; see, e.g., [19, Chapter 5].

On the other hand, solving k-FWA$_{\mathbb{N}}$ turns out to be more difficult. Here, the linear system contains the inequality $w \geq 0$. In fact, it was shown [14] that k-FWA$_{\mathbb{N}}$ is NP-hard. The program Toboggan [14] implements a linear FPT algorithm for MFD$_{\mathbb{N}}$. One step of the algorithm is to solve k-FWA$_{\mathbb{N}}$ using an ILP [14]. The authors state the following conjecture.

▶ **Conjecture 22** ([14]). *If $(P_1, \ldots, P_k)$ are the paths of a minimum flow decomposition of (G, X), then the linear system $L_{\mathbb{N}}(P_1, \ldots, P_k)$ has full rank k.*

As mentioned in the introduction, if the conjecture turned out to be true, then Toboggan could avoid resorting to solving an ILP, since just solving the standard linear system at hand would return its unique solution. As observed by the authors, this would decrease the asymptotic worst case upper bound of Toboggan.

We show that this conjecture is false using a counterexample. Consider the input for k-FWA$_{\mathbb{N}}$ from Figure 4 and the solution therein. We now give another solution for k-FWA$_{\mathbb{N}}$ on this input, namely the following path weights: $a_0 = 5, b_0 = 1$, and $a_i = 6^{2i} + 2, b_i = 6^{2i+1} + 4$, for $i = 1, 2, 3$. One can easily verify that this is another solution to k-FWA$_{\mathbb{N}}$ on the input in Figure 4, thus proving that the rank of the corresponding linear system is strictly less than 8.

To disprove Conjecture 22, it remains to show that any flow decomposition contains at least 8 paths. Due to the technicality of this proof (and its exhaustive case-by-case analysis), in this paper we only explain the intuition behind the construction in Figure 4 and behind the correctness proof. However, as an additional check we also ran both Toboggan [14] and a recently developed ILP solver for MFD$_{\mathbb{N}}$ [7] on this instance, both returning $\mathsf{mfd}_{\mathbb{N}}(G, X) = 8$.

The intuition is as follows. The graph can be divided into two parts: the graph induced by the first 5 vertices in topological order (left part) and the one induced by the last 5 (right part). The exponential growth of the paths A_i and B_i for growing i, together with the different permutations of the paired labels A_iB_j on the left part, fix the choice of the paths A_i and B_i for $i = 1, 2, 3$. This allows us to interpret flow decompositions of less than 8 paths as decompositions with 8 paths, where either A_0 or B_0 carries weight 0. Consider a flow decomposition where we assign two paths of weights λ_1 and λ_2 on the edges labeled A_0B_0. For any $\delta \geq 0$, $(\lambda_1 - \delta) + (\lambda_2 + \delta) = a_0 + b_0$ and equivalently for all other edges on the left part. If we decrease λ_1 by some $\delta > 0$, the weights of B_1 and B_2 each increase by $\delta/2$. And thus, δ must be even. Due to the parity of a_0 and b_0, they can never reach 0.

6 Conclusions

In this paper we have shown for the first time that width, a natural lower bound for MFD, is also useful when investigating its approximability. On the one hand, using width is a key insight in understanding where greedy path removal heuristics fail. On the other hand, graphs where width is well-behaved (e.g., series-parallel graphs) have a guaranteed approximation factor. Moreover, when combined with parity arguments, i.e., about parity fixing unitary flows, and a width-sized decomposition of such flows, it guarantees an even better approximation factor for $\mathrm{MFD}_{\mathbb{Z}}$ for all DAGs. Finally, we have corroborated the complexity gap between the positive integer and the full integer case by disproving a conjecture from [14] (also motivating the heuristic in [20]), which would have had sped up their FPT algorithm for $\mathrm{MFD}_{\mathbb{N}}$.

Our results open up new avenues for further research on MFD. For example, can the width help find larger classes of graphs for which some greedy path removal (or even some sort of greedy *path cover* removal) algorithms have a guaranteed approximation factor? Can we get $\Omega(n)$ worst case approximation ratio of greedy-weight for dense graphs without parallel edges? Can the power-of-two decomposition approach be applied with other factors besides two? Can better path cover-like lower bounds help (e.g., path covers which cannot use an edge more times than its flow value, also computable in polynomial time)? How do our algorithms perform in practice?

References

1 Ravindra K Ahujia, Thomas L Magnanti, and James B Orlin. Network flows: Theory, algorithms and applications. *New Jersey: Prentice-Hall*, 1993.

2 Jasmijn A Baaijens, Leen Stougie, and Alexander Schönhuth. Strain-aware assembly of genomes from mixed samples using flow variation graphs. In *International Conference on Research in Computational Molecular Biology*, pages 221–222. Springer, 2020.

3 Elsa Bernard, Laurent Jacob, Julien Mairal, and Jean-Philippe Vert. Efficient RNA isoform identification and quantification from RNA-Seq data with network flows. *Bioinformatics*, 30(17):2447–2455, 2014.

4 Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Operations Research*, 61(5):1218–1242, 2013. doi:10.1287/opre.2013.1200.

5 Manuel Cáceres, Massimo Cairo, Andreas Grigorjew, Shahbaz Khan, Brendan Mumey, Romeo Rizzi, Alexandru I. Tomescu, and Lucia Williams. Width helps and hinders splitting flows. *CoRR*, 2022. doi:10.48550/ARXIV.2207.02136.

6 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 *SODA 2022 - ACM-SIAM Symposium on Discrete Algorithms*, pages 359–376, 2022. doi:10.1137/1.9781611977073.18.

7 Fernando H. C. Dias, Lucia Williams, Brendan Mumey, and Alexandru I. Tomescu. Fast, Flexible, and Exact Minimum Flow Decompositions via ILP. *arXiv*, arXiv:2201.10923, 2022. To appear in the Proceedings of RECOMB 2022 – 26th Annual International Conference on Research in Computational Molecular Biology. arXiv:2201.10923.

8 Robert P Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950. URL: http://www.jstor.org/stable/1969503.

9 David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992. doi:10.1016/0890-5401(92)90041-D.

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

11 Thomas Gatter and Peter F Stadler. Ryūtō: network-flow based transcriptome reconstruction. *BMC bioinformatics*, 20(1):1–14, 2019.

12 Tzvika Hartman, Avinatan Hassidim, Haim Kaplan, Danny Raz, and Michal Segalov. How to split a flow? In *2012 Proceedings IEEE INFOCOM*, pages 828–836. IEEE, 2012.

13 A. Jain and N. Chandrasekharan. An efficient parallel algorithm for min-cost flow on directed series-parallel networks. In *Proceedings Seventh International Parallel Processing Symposium*, pages 188–192, 1993. doi:10.1109/IPPS.1993.262879.

14 Kyle Kloster, Philipp Kuinke, Michael P O'Brien, Felix Reidl, Fernando Sánchez Villaamil, Blair D Sullivan, and Andrew van der Poel. A practical fpt algorithm for flow decomposition and transcript assembly. In *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 75–86. SIAM, 2018.

15 Brendan Mumey, Samareh Shahmohammadi, Kathryn McManus, and Sean Yaw. Parity balancing path flow decomposition and routing. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2015.

16 Nils Olsen, Natalia Kliewer, and Lena Wolbeck. A study on flow decomposition methods for scheduling of electric buses in public transport based on aggregated time–space network models. *Central European Journal of Operations Research*, 2020. doi:10.1007/s10100-020-00705-6.

17 James B. Orlin. Max flows in O(nm) time, or better. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774. ACM, 2013. doi:10.1145/2488608.2488705.

18 Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature biotechnology*, 33(3):290–295, 2015.

19 Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Inc., 1986.

20 Mingfu Shao and Carl Kingsford. Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(2):658–670, 2017.

21 Mingxiang Teng, Michael I Love, Carrie A Davis, Sarah Djebali, Alexander Dobin, Brenton R Graveley, Sheng Li, Christopher E Mason, Sara Olson, Dmitri Pervouchine, et al. A benchmark for rna-seq quantification pipelines. *Genome biology*, 17(1):1–12, 2016.

22 Alexandru I Tomescu, Travis Gagie, Alexandru Popa, Romeo Rizzi, Anna Kuosmanen, and Veli Mäkinen. Explaining a weighted DAG with few paths for solving genome-guided multi-assembly. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(6):1345–1354, 2015.

23 Alexandru I Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. A novel min-cost flow method for estimating transcript expression with RNA-Seq. In *BMC bioinformatics*, volume 14, pages S15:1–S15:10. Springer, 2013.

24 Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982. doi:10.1137/0211023.

25 Benedicte Vatinlen, Fabrice Chauvet, Philippe Chrétienne, and Philippe Mahey. Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, 185(3):1390–1401, 2008.

26 Lucia Williams, Gillian Reynolds, and Brendan Mumey. RNA Transcript Assembly Using Inexact Flows. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1907–1914. IEEE, 2019.