Width Helps and Hinders Splitting Flows

MANUEL CÁCERES, MASSIMO CAIRO, and ANDREAS GRIGORJEW, Department of Computer Science, University of Helsinki, Helsinki, Finland
SHAHBAZ KHAN, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, India
BRENDAN MUMEY, School of Computing, Montana State University, Bozeman, United States
ROMEO RIZZI, Department of Computer Science, University of Verona, Verona, Italy
ALEXANDRU I. TOMESCU, Department of Computer Science, University of Helsinki, Helsinki, Finland
LUCIA WILLIAMS, Department of Computer Science, University of Montana, Missoula, United States

Minimum flow decomposition (MFD) is the NP-hard problem of finding a smallest decomposition of a network flow/circulation X on a directed graph G into weighted source-to-sink paths whose weighted sum equals X. We show that, for acyclic graphs, considering the *width* of the graph (the minimum number of paths needed to cover all of its edges) yields advances in our understanding of its approximability. For the version of the problem that uses only non-negative weights, we identify and characterise a new class of *width-stable* graphs, for which a popular heuristic is a $O(\log \mathrm{Val}(X))$-approximation ($\mathrm{Val}(X)$ being the total flow of X), 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. We also study a new problem on graphs with cycles, Minimum Cost Circulation Decomposition (MCCD), and show that it generalises MFD through a simple reduction. For the version allowing also negative weights, 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 circulations ($\|X\| \leq 1$), using a generalised notion of width for this problem. Finally, we disprove a conjecture about the linear independence of minimum (non-negative) flow decompositions posed by Kloster et al. [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.

CCS Concepts: • **Theory of computation → Approximation algorithms analysis**; **Network flows**;

Additional Key Words and Phrases: Flow decomposition, graph width

A preliminery version of this paper appeared in the 30th Annual European Symposium on Algorithms, ESA 2022, September 5–9, 2022, Berlin/Potsdam, Germany.
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, 352821, 346968), and partially by the National Science Foundation (NSF) (grants No. 1661530,1759522).
Authors' addresses: M. Cáceres, M. Cairo, A. Grigorjew, and A. I. Tomescu, Department of Computer Science, University of Helsinki, Helsinki 00014, Finland; e-mails: manuel.caceresreyes@helsinki.fi, cairomassimo@gmail.com, andreas.grigorjew@helsinki.fi, alexandru.tomescu@helsinki.fi; S. Khan, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee 247667, India; e-mail: shahbaz.khan@cs.iitr.ac.in; B. Mumey, School of Computing, Montana State University, Bozeman 59717, USA; e-mail: brendan.mumey@montana.edu; R. Rizzi, Department of Computer Science, University of Verona, Verona 37129, Italy; e-mail: romeo.rizzi@univr.it; L. Williams, Department of Computer Science, University of Montana, Missoula 59812, USA; e-mail: lgw2@uw.edu.

ACM 1549-6325/2024/03-ART13
https://doi.org/10.1145/3641820

ACM Reference Format:
Manuel Cáceres, Massimo Cairo, Andreas Grigorjew, Shahbaz Khan, Brendan Mumey, Romeo Rizzi, Alexandru I. Tomescu, and Lucia Williams. 2024. Width Helps and Hinders Splitting Flows. *ACM Trans. Algor.* 20, 2, Article 13 (March 2024), 20 pages. https://doi.org/10.1145/3641820

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 weighted sum equals X. We focus on the case where path weights are restricted to be integers ($\text{MFD}_{\mathbb{Z}}$) or natural numbers ($\text{MFD}_{\mathbb{N}}$). It is a textbook result [Ahujia et al. 1993] that if G is acyclic (a DAG) a decomposition using no more than $m = |E|$ paths always exists. However, MFD is strongly NP-hard [Vatinlen et al. 2008], even on DAGs, and even when the flow values come only from $\{1, 2, 4\}$ [Hartman et al. 2012]. Recent work has shown that the problem is FPT in the size of the minimum decomposition [Kloster et al. 2018] and that it can be formulated as an ILP of quadratic size [Dias et al. 2022].

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 [Bernard et al. 2014; Dias et al. 2023; Gatter and Stadler 2019; Pertea et al. 2015; Tomescu et al. 2015, 2013; Williams et al. 2019] and viral strains [Baaijens et al. 2020]. MFD can also be used to model problems in networking [Hartman et al. 2012; Mumey et al. 2015; Vatinlen et al. 2008] and transportation planning [Olsen et al. 2020], 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. It is known [Hartman et al. 2012] 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, the best known approximation ratio is $\lambda^{\log \|X\|} \log \|X\|$ [Mumey et al. 2015], 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 [1950] and Fulkerson [1956]. 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 to our knowledge. In this article, 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 generalisation of MFD.

We start by considering the connections between the width and a popular heuristic algorithm for $\text{MFD}_{\mathbb{N}}$ which we call *greedy-weight*[1] [Vatinlen et al. 2008], 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., Baaijens et al. [2020], Pertea et al. [2015], and Tomescu et al. [2013] 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

[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.

the greedy-weight approximation factor from $\Omega(\sqrt{m})$ [Hartman et al. 2012], showing for the first time that greedy-weight can be exponentially worse than the optimum:

Theorem 1. *The approximation ratio for greedy-weight on $MFD_{\mathbb{N}}$ is $\Omega(m/\log m)$ for sparse graphs, in the worst case.*

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 identify a new class of graphs, defined by the property that their width does not increase as source-to-sink paths are removed (see Definition 11 of width-stable graphs). We show a relation of width-stable graphs to funnels: precisely, a graph is not width-stable iff it contains a funnel subgraph and a certain *central* path. This is precisely the structure of the class of sparse graphs improving the approximation ratio of greedy-weight in Theorem 1. We also show that width-stability enables greedy-weight to remove paths of large enough flow (Lemma 13), leading to the following result, with $\mathrm{Val}(X)$ being equal to the total flow of the graph:

Theorem 2. *Let $G = (V, E)$ be a width-stable graph and $X : E \to \mathbb{N}$ a flow. Greedy-weight is a $O(\log Val(X))$-approximation for $MFD_{\mathbb{N}}$ on (G, X).*

A notable example of width-stable graphs is the class of *series-parallel graphs*; see [Eppstein 1992; Valdes et al. 1982] 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., Bertsimas et al. [2013] and Jain and Chandrasekharan [1993]). Theorems 1 and 2 show that greedy-weight's approximation ratio is highly linked to the width stability of the graph.

In Section 4, we continue with a generalised version of MFD, **Minimum Cost Circulation Decomposition (MCCD)**, on directed graphs with cycles and no sinks or sources, and a cost function on the edges. Instead of decomposing a flow into weighted paths, we decompose a circulation into *weighted circulations* and minimise the total cost of the circulations, and instead of the width, a natural lower bound for this problem is the **minimum cost of a circulation cover** (**mccc**). Decomposing into circulations rather than paths is a natural generalisation, as paths can be considered as value 1 flows themselves. Additionally, we also consider a relaxation in which the flow/circulation decomposition might use negative integer weights on flows/circulations, rather than strictly positive weights as has traditionally been considered [Hartman et al. 2012; Kloster et al. 2018; Vatinlen et al. 2008]. An important observation that we leverage for this variant (unlike the positive-only version) is that the width/mccc stays constant as flow is chosen and removed. Using this, we give a $(\lceil \log \|X\| \rceil + 1)$-approximation algorithm for this variant.

We denote the problem versions for non-negative path weights and integer path weights by $\mathrm{MCCD}_{\mathbb{N}}$ and $\mathrm{MCCD}_{\mathbb{Z}}$ as well as $\mathrm{MFD}_{\mathbb{N}}$ and $\mathrm{MFD}_{\mathbb{Z}}$, respectively, throughout the article. While $\mathrm{MCCD}_{\mathbb{Z}}$ and $\mathrm{MFD}_{\mathbb{Z}}$ are natural versions of the problem, they have not been previously considered in the MFD literature to our knowledge. However, $\mathrm{MFD}_{\mathbb{Z}}$ can also have natural applications, since by applying $\mathrm{MFD}_{\mathbb{Z}}$ on the difference between two flows, one can minimally explain the *differences* between them, for example, 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 [Teng et al. 2016]. Our approximation follows a *power-of-two* approach, where the weights of the flows/circulations chosen are (positive or negative) powers of two. More specifically, observe that if all circulation values are even, then one can divide them by

2 and obtain a circulation X with smaller $\|X\|$ whose decomposition can be transformed back into a decomposition of X. In order to obtain such an even circulation, we prove a basic property that can be of independent interest: given any integer circulation X, there exists a *unitary* circulation (its values are 0, +1, or −1) Y, such that $X + Y$ is even on every edge (Lemma 21). In addition, given a unitary circulation Y, we show that Y can be decomposed into circulations of total cost no more than mccc (Lemma 23). We obtain the $(\lceil \log \|X\| \rceil + 1)$-approximation ratio (Theorem 3) by iteratively removing the unitary circulation, dividing all circulation values by 2, and preprocessing the graph so that the mccc is a lower bound on the size of the $\text{MCCD}_{\mathbb{Z}}$. Summarised, we show:

Theorem 3. *$MCCD_{\mathbb{Z}}$ can be approximated with a factor of $\log\lceil \|X\| \rceil + 1$ in runtime $O(n \log m(m + n \log n) + m \log \|X\|)$.*

By Corollary 26, we additionally obtain the result for $\text{MFD}_{\mathbb{Z}}$. Notably, the runtime of the algorithm does not depend on the cost function.

Finally, in Section 5, we consider a closely related problem, called *k-Flow Weight Assignment* [Kloster et al. 2018]. 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 Kloster et al. [2018]. 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 Weight 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. Schrijver [1986]). Second, we consider a conjecture from Kloster et al. [2018] stating that if the weights belong to $\mathbb{N}$, and k is the size of a $\text{MFD}_{\mathbb{N}}$ 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 Kloster et al. [2018] for $\text{MFD}_{\mathbb{N}}$, 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 Shao and Kingsford [2017] for $\text{MFD}_{\mathbb{N}}$. In this article, we disprove the conjecture of Kloster et al. [2018], further corroborating the gap between $\text{MFD}_{\mathbb{N}}$ and $\text{MFD}_{\mathbb{Z}}$.

2 PRELIMINARIES

In Sections 3 and 5, we are given a directed acyclic 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. We denote by $\deg^+(v)$ and $\deg^-(v)$ the out- and indegree of a vertex v, respectively. While MFD are also studied for graphs with cycles (see, e.g., Hartman et al. [2012] and Vatinlen et al. [2008]), the task is still to decompose into simple paths, and so our inapproximability result on DAGs in Section 3 also applies for graphs with cycles. In Section 4, we consider directed graphs $G = (V, E, c)$ with no sources or sinks, where $c : E \to \mathbb{R}_{\geq 0}$ is a cost function. Such graphs cannot be acyclic. We use n and m to denote the number of nodes and edges of G, respectively. For both kinds of graphs, we call functions $X : E \to \mathbb{Y}$ *pseudo-flows*,[2] where $\mathbb{Y} \in \{\mathbb{N}, \mathbb{Z}\}$ is some set of allowed flow values. 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

[2]Commonly in the literature, (pseudo-)flows are additionally required to be skew-symmetric and to be upper-bounded by some capacity function on the edges.

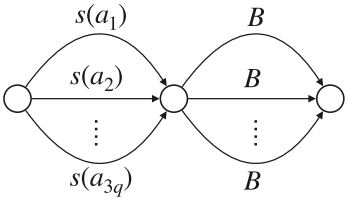


Fig. 1. The reduction of 3-Partition to MFD from Vatinlen et al. [2008]. The 3-Partition instance consists of a set $A = \{a_1, \ldots, a_{3q}\}$, where every a_i has a positive integer *size* $s(a_i)$, and a positive integer B, such that $B/4 < s(a_i) < B/2$ holds for every $a_i \in A$. The question is whether A can be partitioned into q disjoint subsets, each of 3 elements and of size B. The MFD series-parallel (see Definition 15) reduction consists of a subgraph obtained by the *parallel composition* of $3q$ edges with flow values $s(a_1), \ldots, s(a_{3q})$, and a subgraph obtained by the parallel composition of q edges, each with flow value B. These two graphs are composed with the *series composition*. Intuitively, because $B/4 < s(a_i) < B/2$ holds for every a_i, the MFD consists of exactly $3q$ paths of weights $s(a_1), \ldots, s(a_{3q})$, and each edge on the right-hand subgraph is traversed by exactly three paths whose weights sum to B, giving thus the partition of A. Moreover, since the first $3q$ edges need to be decomposed, the previous decomposition is minimum even if negative weights are allowed, making $\text{MFD}_{\mathbb{Z}}$ NP-hard.

multiplication by a scalar, respectively. The numbers 0 and 1 also denote (depending on context) pseudo-flows that are 0 (resp. 1) on every edge. We write $X \leq Y$ (and similarly $<$) to mean $X(u,v) \leq Y(u,v)$ for every $(u,v) \in E$.

Given a DAG G, a *flow* is a pseudo-flow satisfying conservation of flow (incoming flow equal to outgoing flow) on internal nodes $V \setminus \{s,t\}$. A pseudo-flow satisfying the conservation of flow on all nodes is called a *circulation*. We sometimes refer to the value $X(e)$ for a flow/circulation X as the flow of the edge e. It is known that the sum of two flows/circulations $X + Y$, the multiplication of a flow/circulation with by a scalar aX, and the empty pseudo-flow 0 are themselves flows/circulations. For a flow X, let $\text{Val}(X)$ denote the total flow out of s (or equivalently into t by flow conservation). Note that $\text{Val}(X)$ can be negative. Given an s-t path P, denote by $P[u..v]$ the subpath of P going from $u \in V$ to $v \in V$ and let P also denote the flow defined by setting *one* to every edge in P and 0 to every other edge. With these definitions, we are ready to formally define MFD.

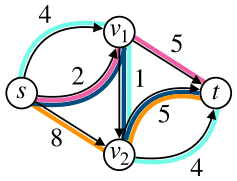
Definition 4. Given a flow X, a *flow decomposition* of (G,X) of *size* k is a family of s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ with weights $(w_1, \ldots, w_k) \in \mathbb{Y}^k$ such that $X = w_1P_1 + \cdots + w_kP_k$.

Definition 5. For a flow X, let $\text{mfd}_{\mathbb{Y}}(G,X)$ be the smallest size of a flow decomposition of (G,X) with weights in $\mathbb{Y}$.

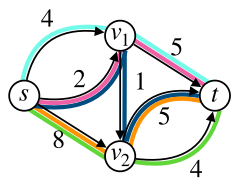
We omit $\mathbb{Y}$ if it is clear from the context. We call the problem of producing a flow decomposition of (G,X) of minimum size the *MFD problem*.

In Section 3, we study $\text{MFD}_{\mathbb{N}}$ ($0 \in \mathbb{N}$), and in Section 4, we study $\text{MFD}_{\mathbb{Z}}$ and its generalisation $\text{MCCD}_{\mathbb{Z}}$. Note that the reduction showing $\text{MFD}_{\mathbb{N}}$ to be strongly NP-hard from Vatinlen et al. [2008] also holds for $\text{MFD}_{\mathbb{Z}}$ (see Figure 1). However, a flow with non-negative values may admit a decomposition using fewer paths if negative weights are allowed, as shown in Figure 2. We explore further differences between $\text{MFD}_{\mathbb{N}}$ and $\text{MFD}_{\mathbb{Z}}$ in Sections 4 and 5.

Let $\|X\| = \max_{(u,v)\in E} |X(u,v)|$ denote the maximum norm on flows or circulations. 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 $X \equiv_2 Y$ iff X and Y have the same parity everywhere, that is, for every $(u,v) \in E$, we have that $X(u,v)$ is odd iff $Y(u,v)$ is odd.



(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.

Fig. 2. A positive flow admitting a decomposition into four paths only if negative weights are allowed.

Definition 6. Given $S \subseteq E$, we define $\text{width}_S(G)$ as the minimum number of s-t paths in a DAG G needed to cover all edges of S. If $S = E$ we just write $\text{width}(G)$.

The width is the main combinatorial tool that we use for our approximation results, and we will show in Section 3, that it is highly linked to the approximation performance of greedy-weight. Just like its more common node variant, $\text{width}(G)$ can be computed in $O(mn)$ time. As described by, for example, Ahujia et al. [1993] and Cáceres et al. [2022], this is done by reduction to a min-flow instance with demand one on every edge; the minimum flow of this instance is $\text{width}(G)$, 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 7 (Ahujia et al. [1993] and Orlin [2013]). *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 $|C| = \text{width}_S(G)$.*

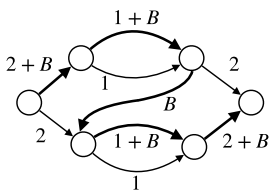
The flow C with total flow $\text{width}_S(G)$ suffices and we do not need to calculate a path cover achieving that minimum. However, we note that it can be directly computed given the flow C. We can think of this path cover as a flow decomposition of C into $\text{width}_S(G)$ weight-one paths, which 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 $\text{width}_S(G) \leq m$, the overall runtime of finding the path cover is $O(mn)$. Similarly, every path cover $P_1, \ldots, P_\ell$ of G defines a flow C on G: $C = P_1 + \cdots + P_\ell$, and we say that C is the induced flow of the path cover $(P_i)_1^\ell$.

Definition 8. In a directed graph $G = (V, E)$ we call a subset $C \subseteq E$ an *antichain* of G if all edges in C are pairwise parallel, that is there exists for no pair of edges in C a path in G leading from one edge to the other.

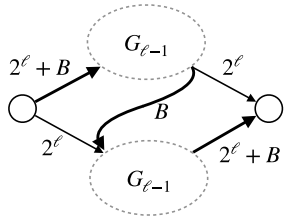
It can be shown with straight forward arguments that $\text{width}(G) = |C|$ for a maximum (sized) antichain C of G.

3 WIDTH MATTERS FOR GREEDY APPROACHES

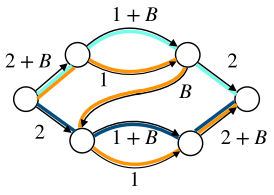
Since the difference of two flows is still a flow, it is very natural to consider successively removing the simplest type of flow — that is to say, paths — as an algorithmic strategy for $\text{MFD}_{\mathbb{N}}$. Indeed, the particular greedy path removal strategy of finding a *heaviest path* (*greedy-weight*) is commonly used as a heuristic in applications (e.g., Baaijens et al. [2020], Hartman et al. [2012], Pertea et al. [2015], and Tomescu et al. [2013]) and it seems to be mentioned in nearly every article 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 (in particular, a v-v path carries infinite flow). A *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., Vatinlen et al. [2008]).



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



(b) Building $(G_\ell, X_{\ell,B})$ from two copies of $(G_{\ell-1}, X_{\ell-1,B})$ $(\ell > 1)$. The 5 edges entering (resp. leaving) $(G_{\ell-1}, X_{\ell-1,B})$ are defined to enter (resp. leave) the source (resp. sink) node 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.

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

3.1 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 3(a). For $\ell > 1$, we build $(G_\ell, X_{\ell,B})$ from two disjoint copies of $(G_{\ell-1}, X_{\ell-1,B})$, by adding five extra edges and flow values as shown in Figure 3(b). 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. By 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 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.

Lemma 9. *For the graph G_ℓ with flow $X_{\ell,2^{\ell+1}}$, greedy-weight uses $1 + 2^{\ell+1}$ paths to decompose $X_{\ell,2^{\ell+1}}$.*

Proof. We first show that the heaviest path in $X_{\ell,2^{\ell+1}}$ follows every backbone edge from s to t. Certainly this path carries $2^{\ell+1}$ flow, and no more, since every backbone edge has flow at least $2^{\ell+1}$ and every central edge has flow exactly $2^{\ell+1}$. To see that there is no heavier path, observe that all the non-backbone edges have flow value strictly less than $2^{\ell+1}$ by construction. After removing that path with weight $2^{\ell+1}$, all the central edges are completely decomposed. The remaining graph and flow, without central edges, has $2^{\ell+1}$ weight-1 edges that are pairwise non-reachable using only non-zero flow edges (four edges for each of the $2^{\ell-1}$ copies of G_1), each of which must be covered by a different path of weight 1 (and these paths fully decompose the flow). □

Lemma 10. *For the graph G_ℓ, flow $X_{\ell,2^{\ell+1}}$ can be decomposed using $2\ell + 2$ paths.*

Proof. Using induction, we first prove that we can use $2\ell + 1$ paths to decompose all of the flow of $X_{\ell,2^{\ell+1}}$ on the non-backbone edges of G_ℓ, and that these paths have total weight $2^{\ell+2} - 3$. When $\ell = 1$ (see Figure 3(c)), we can decompose both the flow-1 edges with a single path of weight 1, which goes through the central edge. Moreover, we can decompose the flow-2 edges with two paths of weight 2, without using the central edge. These paths have $5 = 2^{1+2} - 3$ total flow. We now assume that the claims hold for $\ell = k$ and prove it for $\ell = k + 1$. Consider the graph G_{k+1}. By assumption, the non-backbone edges in every copy of G_k are fully decomposed by $2k + 1$ paths, and the total flow of those paths is $2^{k+2} - 3$. Note that these paths are s-t paths in G_k, but they must

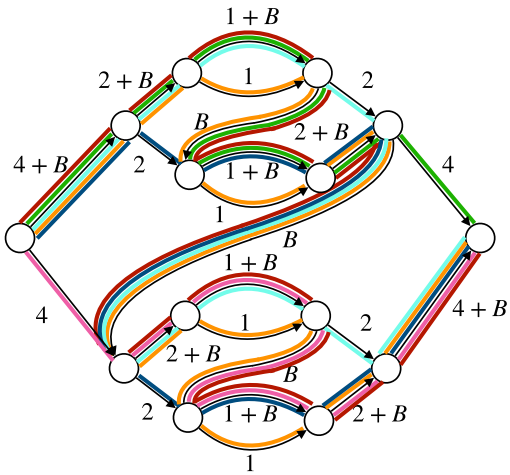


Fig. 4. $(G_2, X_{2,B=2^{\ell+1}})$ with a $2\ell + 2 = 6$ flow decomposition. Orange (weight 1) and dark and light blue (weight 2) decompose the non-backbone edges in the two copies of G_1. Green and pink (weight 4) decompose the non-backbone edges added in G_2. Dark red (weight 3) is the additional path needed to fully decompose the flow.

be extended to be s-t paths in G_{k+1}. Because there is a central edge with weight $2^{(k+1)+1} = 2^{k+2} > 2^{k+2} - 3$ from one copy of G_k to the other, it is possible to route the $2k + 1$ paths from the first G_k to the second using the central edge. Additionally, the backbone edges from s to the first copy of G_k and from the second copy of G_k to t have flow $2^{k+1} + 2^{k+2}$, so they can be used to complete the routing of the paths from s to t. Then we can use two additional paths of weight 2^{k+1} each to decompose the flow of the two new non-backbone edges; one using the backbone path of the upper G_k (extended by the upper edges of G_{k+1}), and analogously, the other through the backbone path of the lower G_k, which is possible since the paths obtained inductively only decompose 2^{k+1} of the backbone flow of G_k whereas now 2^{k+2} backbone flow must be decomposed. As such, we use $2k + 1 + 2 = 2(k + 1) + 1$ paths with total flow $2^{k+2} - 3 + 2(2^{k+1}) = 2^{(k+1)+2} - 3$, as required. By the previous, given any graph G_ℓ, we can decompose all non-backbone edges using $2\ell + 1$ paths. Note that removing these weighted paths from $X_{\ell,2^{\ell+1}}$ yields a flow on G_ℓ (of value 3). Because the remaining edges (all backbone) form a path from s to t, and the remaining edge values form a flow on G_ℓ, all remaining edges must have the same flow value, and can be covered by one path. Thus, $2\ell + 2$ paths are sufficient to decompose $X_{\ell,2^{\ell+1}}$. See Figure 4 for an example when $\ell = 2$. □

Theorem 1. *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 9 and 10, 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 the 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 [Vatinlen et al. 2008]: 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 [Baaijens et al. 2020]) is the *longest path* (with its maximum possible flow so that it is saturating). To adapt our construction

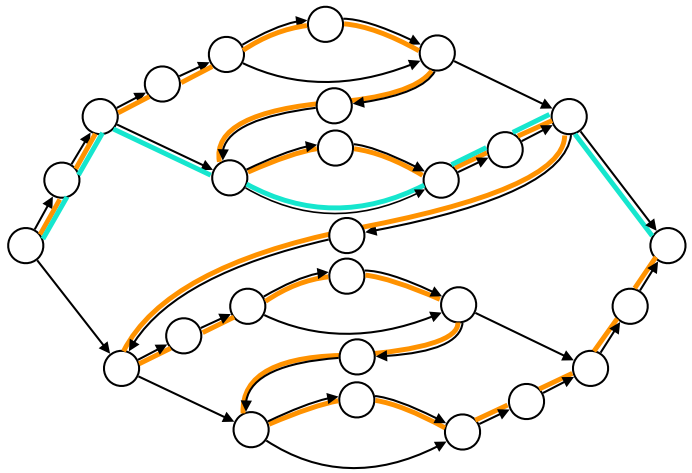


Fig. 5. By subdividing the backbone edges, choosing longest paths (orange) and shortest paths (blue) both give the same approximation ratio as choosing heaviest paths in the original flow $X_{\ell,2^{\ell+1}}$.

of $(G_\ell, X_{\ell,2^{\ell+1}})$ so that this approach yields 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 split every backbone edge (u, v) into two edges, (u, w) and (w, v). See Figure 5 for an example. Then the path along the backbone edges will be the longest from s to t and the previous asymptotic analysis still holds, since we no more than doubled the number of edges (and the number of edges of the new construction is still $\Theta(2^\ell)$). Yet another optimality criterion, studied in Hartman et al. [2012] 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 decompose all $2^{\ell+1}$ weight-1 edges with $2^{\ell+1}$ different paths.

3.2 Greedy Approximation for Width-Stable Graphs

As exploited in Section 3.1, 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 identify a new class of graphs, in which the graph does not increase its width during the execution of the algorithm. We show that greedy-weight decomposes "enough" flow at each step in these graphs, giving a $O(\log \mathrm{Val}(X))$-approximation for $\mathrm{MFD}_{\mathbb{N}}$.

If $X \geq 0$ is a flow on a DAG 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 $\mathrm{MFD}_{\mathbb{N}}$, 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 *width-stable* if it does not have this issue.

Definition 11 (Width-stable Graph). We say that a graph G is *width-stable* if, for any non-negative flows $X \leq Y$ on G, it holds that $\mathrm{width}(G|_X) \leq \mathrm{width}(G|_Y)$.

Many useful $\mathrm{MFD}_{\mathbb{N}}$ instances satisfy Definition 11. For example, the first proof of MFD's NP-hardness [Vatinlen et al. 2008] was a reduction to a very simple graph of this form, as shown in Figure 1; this means that $\mathrm{MFD}_{\mathbb{N}}$ restricted to width-stable graphs is also NP-hard.

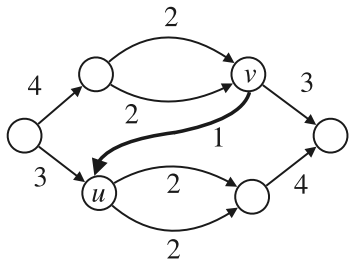


Fig. 6. $\mathrm{MFD}_{\mathbb{N}}$ instance with $\mathrm{width}(G) = 3$, $\mathrm{Val}(X) = 7$, $\mathrm{Val}(X)/\mathrm{width}(G) = 7/3 > 2$, but no path can carry more flow than 2. By Lemma 13, this is equivalent to G not being width-stable, there exists an s-t path saturating the central edge from v to u of weight 1, making the graph a funnel and increasing $\mathrm{width}(G)$ to 4.

Definition 12 (Garlet Millani et al. [2020]). We call an s-t DAG G *funnel* if every s-t path has a private edge that is not contained in any other s-t path.

Funnels are simple graphs in the sense that they admit a unique flow decomposition [Khan et al. 2022]. We use funnels in Lemma 13 to characterize graphs that are not width-stable. Funnels generalise in/out-forests: along any s-t path nodes v first satisfy $\deg^-(v) \leq 1 \leq \deg^+(v)$ and then $\deg^-(v) \geq 1 \geq \deg^+(v)$. We call a node v *forking* (resp. *merging*) if $\deg^+(v) > 1$ (resp. if it is $\deg^-(v) > 1$). For a funnel subgraph F of G we call a path in G from a merging node in F to a forking node in F a *central path* of the funnel. The graphs $(G_\ell, X_{\ell,B})$ in Section 3.1 are precisely funnels with central paths.

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

Lemma 13. *Let G be an s-t DAG. The following statements are equivalent:*

(1) G is width-stable,
(2) G has paths of large weight*: for any flow $X \geq 0$ on G, there exists an s-t path in $G|_X$ carrying $\mathrm{Val}(X)/\mathrm{width}(G|_X)$ flow,*
(3) G has no funnel subgraph with a central path.

See Figure 6 for an example.

Proof. (1) $\Rightarrow$ (2): Let $G = (V, E)$ be an s-t DAG with flow $X \geq 0$ for which there is no s-t path in $G|_X$ carrying $b = \mathrm{Val}(X)/\mathrm{width}(G|_X)$ flow. We will show that G is not width-stable. For simplicity, we assume $G = G|_X$, the result follows immediately for any supergraph of G. Let S be the set of vertices reachable from s by a path carrying b flow. By assumption, $t \notin S$, so $(S, V \setminus S)$ is an s,t-cut and $X(u,v) < b$ for any edge (u,v) with $u \in S, v \in V \setminus S$. Define T to be the set of vertices that can reach t via a path involving vertices only from $V \setminus S$:

$$T := V \setminus \{v \in V \mid \text{all } v - t \text{ paths cross } S\}.$$

Since $S \subseteq V \setminus T$, $(V \setminus T, T)$ is an s,t-cut. Note that if (u,v) is an edge with $u \in V \setminus T$ and $v \in T$, then also $u \in S$ and $v \in V \setminus S$ and thus $X(u,v) < b$. The set C defined by

$$C := \{(u,v) \in E \mid u \in V \setminus T, v \in T\} = E \cap (S \times T)$$

is an s,t-cut set (a set of edges that every s-t path has to cross) and we have $X(e) < b$ for every $e \in C$. This implies $|C| > \mathrm{width}(G)$.

We construct a new flow $Y \geq 0$ on G for which C is an antichain in $G|_Y$. Initially, $Y := X$. Define the following paths:

— $p_s(u)$ for all $u \in S$: an s-u path with all nodes in S,
— $p_t(v)$ for all $v \in T$: a v-t path with all nodes in T.

We keep the **invariant** that the paths exist and do not change in $G|_Y$ throughout the construction of Y. Let $b(v,u)$ denote a path from $v \in T$ to $u \in S$, all of whose internal nodes are from $V \setminus (S \cup T)$, and note that such a path exists iff C is not an antichain. Assume it carries flow $\mu > 0$ (i.e., μ is the minimum flow value along $b(v,u)$), we then perform the following operation on Y:

$$Y(e) = \begin{cases} Y(e) - \mu & \text{if } e \in b(v,u) \\ Y(e) + \mu & \text{if } e \in p_s(u) \cup p_t(v). \end{cases}$$

Note that after the operation Y remains a flow and that none of the three paths have pairwise intersecting edges. The process does not violate the **invariant** and repeating eventually it destroys all paths from T to S, making C an antichain in $G|_Y$. This shows that G is not width-stable: $Y \le X + Y$ and $\mathsf{width}(G|_Y) > \mathsf{width}(G|_{X+Y}) = \mathsf{width}(G)$.

(2) $\Rightarrow$ (3): Assume that G has a funnel subgraph F with a central path P. Let C be a maximum antichain of F, and note that every maximum antichain of a funnel consists of private edges only. We define flows Z_{stable}, Z_{unstable}, and Z:

— Z_{stable} is defined to be the flow induced[3] by the minimum path cover of F[4] (i.e., $G|_{Z_{\text{stable}}} = F$).
— Z_{unstable} is defined to be the flow induced by the following path cover of the graph consisting of F and P: One s-t path goes along P, covering two edges in C,[5] and the other paths cover F avoiding P, this is possible with additional $|C| - 2$ paths.
— Finally, $Z := Z_{\text{stable}} + Z_{\text{unstable}}$.

We have $\mathsf{Val}(Z) = \mathsf{Val}(Z_{\text{stable}}) + \mathsf{Val}(Z_{\text{unstable}}) = |C| + (|C| - 1) = 2|C| - 1$ and $\mathsf{width}(G|_Z) \le |C| - 1$, and thus

$$\frac{\mathsf{Val}(Z)}{\mathsf{width}(G|_Z)} \ge \frac{2|C| - 1}{|C| - 1} > 2,$$

but all s-t paths in G carry no more flow than 2, because C is an s,t-cut set of G and all Z flow values on C are 2.

(3) $\Rightarrow$ (1): Assume that $G = (V,E)$ is not width-stable, let $Y \ge X \ge 0$ be flows on G with $\mathsf{width}(G|_X) > \mathsf{width}(G|_Y)$ and let C' be a maximum antichain of $G|_X$. Let F be a funnel subgraph of $G|_X$ containing C' as maximum antichain, and let C be the rightmost maximum antichain of F (that is, the head of every edge in C is t or is merging). Since C is not an antichain of $G|_Y$, there must be a path P in $G|_Y$ connecting two edges in C, and it starts at a merging node. It must also enter a forking node, because otherwise adding the path would not decrease the width. This shows that a prefix path of P is a central path of F. □

Lemma 14. *Let $G = (V,E)$ be a width-stable graph, width$(G) \ge 2$. Greedy-weight uses at most $\lfloor \log \mathit{Val}(X) / \log \frac{\mathit{width}(G)}{\mathit{width}(G)-1} \rfloor + 1$ paths to decompose any flow $X : E \to \mathbb{N}$.*

Proof. Let $b = \mathsf{width}(G)$. Since G is width-stable, greedy-weight removes a path of weight at least $\mathsf{Val}(X')/b$ at every step by Lemma 13, where X' is the remaining flow of the corresponding step. As such, after c steps $\mathsf{Val}(X') \le \mathsf{Val}(X)(\frac{b-1}{b})^c$. If $\mathsf{Val}(X)(\frac{b-1}{b})^c < 1$, then $\mathsf{Val}(X') = 0$, since $\mathsf{Val}(X)$ and the weights of the removed paths belong to $\mathbb{N}$. Solving for c we obtain $c >$

[3]Recall that the induced flow X of a path cover $(P)_1^\ell$ is defined as $X = P_1 + \cdots + P_\ell$, where we identify each path P_i with a 0/1-flow with value 1 on the path edges (i.e., precisely the characteristic function of P_i).
[4]Since funnels admit a unqiue flow decomposition, they also admit a unique minimum s-t path cover.
[5]Indeed, this path covers *exactly* two edges in C: one edge needs to be covered to reach P and another edge must be covered to reach t, and since G is a DAG, we cannot use the same edge twice.

$\log \mathrm{Val}(X)/\log \frac{b}{b-1}$. Therefore, greedy-weight takes (uses) at most $c = \lfloor \log \mathrm{Val}(X)/\log \frac{b}{b-1} \rfloor + 1$ steps (paths). □

Theorem 2. *Let $G = (V, E)$ be a width-stable graph and $X : E \to \mathbb{N}$ a flow. Greedy-weight is a $O(\log \mathit{Val}(X))$-approximation for $\mathit{MFD}_{\mathbb{N}}$ on (G, X).*

Proof. Assume $X > 0$ (otherwise, replace G by $G|_X$). Thus, $b = \mathrm{width}(G) \leq \mathrm{mfd}_{\mathbb{N}}(G, X)$, 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 14 implies that greedy-weight is a $O(\frac{\log \mathrm{Val}(X)}{b \log \frac{b}{b-1}}) = O(\log \mathrm{Val}(X))$-approximation for $\mathrm{MFD}_{\mathbb{N}}$ ($b \log \frac{b}{b-1} = O(1)$ for $b \geq 2$). □

Finally, we show that series-parallel graphs are width-stable, and thus greedy-weight is a $O(\log \mathrm{Val}(X))$-approximation on them.

Definition 15 (Series-parallel Graph [Eppstein 1992]). 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 16. *Greedy-weight is a $O(\log \mathit{Val}(X))$-approximation for $\mathit{MFD}_{\mathbb{N}}$ on series-parallel graphs.*

Proof. Using Theorem 2, it remains to prove that any series-parallel graph $G = (V, E)$ with any flow $X : E \to \mathbb{N}$ is width-stable. We prove it using structural induction. The base case is when G is single edge from s to t, and they are trivially width-stable.

Suppose now that G is obtained by the composition of series-parallel graphs G_1, G_2, and let $X \leq Y$ by any non-negative flows on G. Let $X_i = X|_{G_i}$, $Y_i = Y|_{G_i}$ and let x_i denote $\mathrm{width}(G_i|_{X_i})$, y_i denote $\mathrm{width}(G_i|_{Y_i})$, for $i = 1, 2$. $x_i \leq y_i$ for $i = 1, 2$.

If the composition operation is parallel composition, then $\mathrm{width}(G) = \mathrm{width}(G_1) + \mathrm{width}(G_2)$ (since edges of G_1 cannot reach edges of G_2, and vice versa) and $\mathrm{width}(G|_X) = x_1 + x_2$, $\mathrm{width}(G|_Y) = y_1 + y_2$ (since $G|_X$ and $G|_Y$ are also series-parallel), and G is width-stable by the inductive hypothesis that $x_i \leq y_i$, for $i = 1, 2$.

If the composition operation is series composition, width-stability follows analogously to the parallel composition by replacing sum with maximum. □

However, note that there are width-stable graphs that are not series-parallel.

4 WIDTH HELPS SOLVE $\mathrm{MCCD}_{\mathbb{Z}}$

In this section, we give an approximation algorithm for $\mathrm{mfd}_{\mathbb{Z}}(G, X)$. We will obtain this for a more general problem variant, which can be defined as follows. We are given directed a graph $G = (V, E, c)$ with no sink or source nodes, with (cost) a function $c : E \to \mathbb{R}_{\geq 0}$. The *cost* of a circulation X is defined as $\tilde{c}(X) = \sum_{e \in E} c(e)X(e)$. Note that $\tilde{c}(\cdot)$ is a linear function: $\tilde{c}(X + Y) = \tilde{c}(X) + \tilde{c}(Y)$ for any two circulations X, Y on G.

Definition 17. Given (G, X) of a graph $G = (V, E, c)$ and a circulation $X : E \to \mathbb{Y}$, a *circulation decomposition* of size k of (G, X) is a family of circulations $Y_i : E \to \mathbb{N}$ with weights $(w_1, \ldots, w_k) \in \mathbb{Y}^k$ such that $X = w_1 Y_1 + \cdots + w_k Y_k$. We call the problem of finding a circulation decomposition

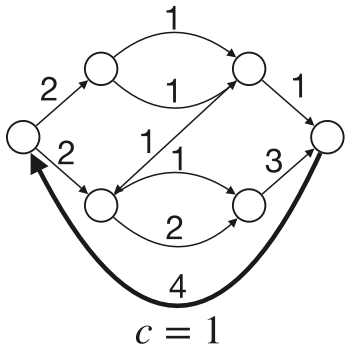


Fig. 7. Reduction in Lemma 18 of a graph with total flow 4 from $\text{MFD}_{\mathbb{Y}}$ to $\text{MCCD}_{\mathbb{Y}}$. The bold edge is the addition to the DAG and is the only edge with cost 1 while all other edges have cost 0.

of minimum cost $\tilde{c}(Y_1 + \cdots + Y_k)$ the *Minimum Cost Circulation Decomposition* or $\text{MCCD}_{\mathbb{Y}}$ and we write $\text{mccd}_{\mathbb{Y}}(G, X)$ for the minimum cost.

Decomposing into *non-negative* weighted circulations rather than paths is a natural generalisation, as paths can also be seen as flows with value 1 along the path. The following reduction (Figure 7) shows that $\text{MFD}_{\mathbb{Y}}$ can be regarded as a special case of $\text{MCCD}_{\mathbb{Y}}$.

Lemma 18. *$\text{MCCD}_{\mathbb{Y}}$ is NP-hard.*

Proof. Given an s-t DAG $G = (V, E)$ and flow $X : E \to \mathbb{Y}$, we define a graph $G' = (V, E', c)$ with $E' = E \cup \{(t, s)\}$ and $c : E' \to \{0, 1\}$, $c(u, v) = 1 \iff (u, v) = (t, s)$, that is, cost 1 only for the edge (t, s). Let $X' : E' \to \mathbb{Y}, X'(e) = X(e)$ for $e \in E$ and $X'(t, s) = \text{Val}(X)$ be a circulation on G'. The cost of a MCCD $(Y'_1, w'_1), \ldots, (Y'_k, w'_k)$ of (G', X') is equal to $\tilde{c}(Y'_1) + \cdots + \tilde{c}(Y'_k) = Y'_1(t, s) + \cdots + Y'_k(t, s)$. Note that an s-t path $P : E \to \{0, 1\}$ yields a circulation $Y_P : E' \to \{0, 1\}$ of cost 1, which implies $\text{mfd}_{\mathbb{Y}}(G, X) \geq \text{mccd}_{\mathbb{Y}}(G', X')$. Define flows $Y_i : E \to \mathbb{Y}$ with $Y_i(e) = Y'_i(e)$ for $i = 1, \ldots, k$. Decomposing each flow Y_i trivially into $\text{Val}(Y_i) = Y'_i(t, s)$ paths and assigning them weights w'_i yields a Flow Decomposition of (G, X), showing $\text{mccd}_{\mathbb{Y}}(G', X') \geq \text{mfd}_{\mathbb{Y}}(G, X)$, and thus this Flow Decomposition is minimum. □

Definition 19. Given $S \subseteq E$ and a graph G, we call a circulation C of minimum cost satisfying $C(e) \geq 1$ for all $e \in S$ Minimum Cost Circulation Cover of S, and we write $\text{mccc}_S(G) = \tilde{c}(C)$. If $S = E$, we use $\text{mccc}(G)$ instead.

Note that since every circulation decomposition of a graph G covers the edges of non-zero circulation, $\text{mccc}_S(G) \leq \text{mccd}_{\mathbb{Y}}(G, X)$ with $S = \{e \in E \mid X(e) \neq 0\}$. This generalises the width lower bound of $\text{MFD}_{\mathbb{Y}}$ to $\text{MCCD}_{\mathbb{Y}}$. Given an s-t DAG G, $\text{width}(G) = \text{mccc}(G')$ for the graph G' obtained by the reduction in Lemma 18. We will need a Minimum Cost Circulation Cover for our approximation approach:

Lemma 20 (Gabow and Tarjan [1989], Theorem 3.6). *Let $G = (V, E, c)$ be a graph, and $S \subseteq E$. A Minimum Cost Circulation Cover of S can be computed in $O(n \log m(m + n \log n))$ time.*

The idea behind our approximation algorithm for $\text{MCCD}_{\mathbb{Z}}$ is that a circulation $X : E \to \mathbb{Z}$ on a graph G can always be decomposed into circulations of total cost $(\lceil \log \|X\| \rceil + 1) \cdot \text{mccc}(G)$. We show this using two key facts: first, that X can be decomposed into $(\lceil \log \|X\| \rceil + 1)$ circulations with a particular structure, and, second, that each of these circulations can be further decomposed into circulations of total cost of at most $\text{mccc}(G)$. A key step in proving both these facts is a subroutine which, given an input circulation X, finds another circulation Y with values from $\{0, \pm 1\}$

only (a *unitary* circulation) that matches the parity of X on all edges. Intuitively, given an input circulation X, such a unitary circulation Y can be added to X to "fix" its odd edges to be even, with only a small change to X.

Lemma 21. *For any circulation $X : E \to \mathbb{Z}$ on $G = (V, E, c)$, there exists a circulation $Y : E \to \mathbb{Z}$ 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} has even degree due to the conservation of flow. Thus, G_{odd} can be written as the edge-disjoint union of cycles. Assign an arbitrary orientation to each cycle 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 circulation decomposed as a sum of circulations, each along one of the edge-disjoint cycles. Moreover, $X \equiv_2 Y$ and $\|Y\| \leq 1$ by construction. □

Repeatedly applying Lemma 21 and dividing the resulting even circulation by 2, we obtain the the first key ingredient of the approach.

Corollary 22. *Any (non-zero) circulation $X : E \to \mathbb{Z}$ 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 circulation with $\|Y_i\| \leq 1$ for all i.*

Proof. If $\|X\| \leq 1$ the result follows. Otherwise apply Lemma 21 to obtain Y_0 such that $X \equiv_2 Y$ and $\|Y_0\| \leq 1$. Since $X \equiv_2 Y_0$, we can define $X' = (X - Y_0)/2$, and thus $X = 2X' + Y_0$. Recursively repeat this procedure on X' until $\|X'\| \leq 1$, obtaining $Y_0, \ldots, Y_k = X'$, so that $X = \sum_{i=0}^{k} 2^i \cdot Y_i$. Finally, note that at each repetition, $\|X\|$ decreases to at most $\lceil \|X\|/2 \rceil$, thus $k \leq \lceil \log \|X\| \rceil$. □

The following result is the second key ingredient of our approach. It guarantees that any unitary circulation can be decomposed into two circulations of total cost of at most mccc(G) (see Figure 8 for an example). This is by no means obvious since, among other problems, a unitary circulation may contain positive and negative values which merge and cancel each other out (as in Figure 8(b)).

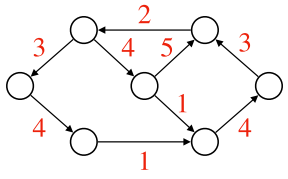
Lemma 23. *For any circulation $X : E \to \mathbb{Z}$, $\|X\| \leq 1$, there exist circulations $A, B : E \to \mathbb{Z}$ such that:*

(1) $A, B \geq 0$
(2) $X = A - B$
(3) $\tilde{c}(A) + \tilde{c}(B) \leq mccc(G)$

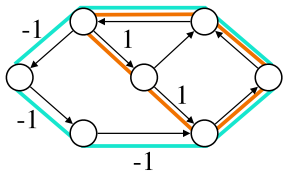
Proof. Take C such that $C \geq 1$ and $\tilde{c}(C) = \text{mccc}(G)$, according to Lemma 20. Take D such that $D \equiv_2 X + C$ and $\|D\| \leq 1$, according to Lemma 21. Also, assume $\tilde{c}(D) \geq 0$ without loss of generality (otherwise, take $-D$, which satisfies the same properties).

Since $D \equiv_2 X + C$, we have $C - D \pm X \equiv_2 0$. We now consider the circulations $A := (C - D + X)/2$ and $B := (C - D - X)/2$, which have the following properties:

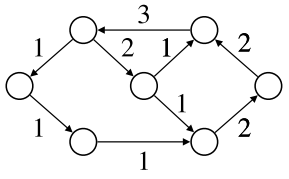
(1) Notice that $C - D \pm X \geq C - 2$ since $\|D\|, \|X\| \leq 1$. So, $C - D \pm X \geq -1$, since $C \geq 1$. But $C - D \pm X \equiv_2 0$ so $C - D \pm X \geq 0$, therefore $A, B \geq 0$.
(2) $A - B = \frac{C-D+X}{2} - \frac{C-D-X}{2} = X$.



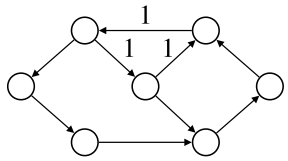
(a) Edge costs



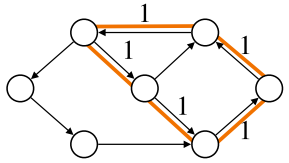
(b) Unitary circulation X on a graph G and a decomposition into two non-negative circulations A and B, of weight 1 in orange (see **(e)**), and of weight -1 in blue (see **(f)**), respectively.



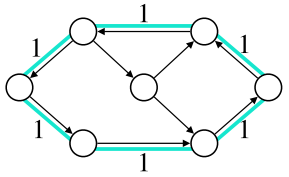
(c) Circulation C covering all edges of G, of cost $\tilde{c}(C) = \text{mccc}(G) = 42$ (Lemma 20).



(d) Unitary circulation D of cost 11 matching the parity of $X + C$, i.e., $D \equiv_2 X + C$ (Lemma 21).



(e) Circulation $A = (C - D + X)/2$ of cost 14.



(f) Circulation $B = (C - D - X)/2$ of cost 17.

Fig. 8. Example of Lemma 23 applied to a unitary circulation X on a graph G (for clarity, 0 circulation values are not shown). Non-negative circulations A and B can be constructed so that $\tilde{c}(A) + \tilde{c}(B) \leq \text{mccc}(G)$ holds. We obtain a decomposition of X by $X = A - B$.

(3) $\tilde{c}(A) + \tilde{c}(B) = \tilde{c}(A + B) = \tilde{c}(\frac{C-D+X}{2} + \frac{C-D-X}{2}) = \tilde{c}(C - D) = \tilde{c}(C) - \tilde{c}(D) \leq \tilde{c}(C)$ since $\tilde{c}(D) \geq 0$, and $\tilde{c}(C) = \text{mccc}(G)$. □

Finally, expressing any circulation as a sum of at most $\lceil \log \|X\| \rceil + 1$ unitary circulations (Corollary 22), and decomposing each unitary circulation into two circulations with cost of at most $\text{mccc}(G)$ (Lemma 23), we can decompose the circulation into circulations of total cost no more than $(\lceil \log \|X\| \rceil + 1) \cdot \text{mccc}(G)$ whose weights are positive and negative powers of two.

Theorem 24. *Given a graph $G = (V, E, c)$ and a circulation $X : E \to \mathbb{Z}$ with $k := \lceil \log \|X\| \rceil$, there exist circulations A_i, B_i for $i = 0, \ldots, k$ and weights $\{w_0, \ldots, w_k\} \subseteq \{2^i \mid i \in \mathbb{N}\}$, with $\tilde{c}(A_0 + \cdots + A_k + B_0 + \cdots + B_k) \leq (k + 1) \cdot \text{mccc}(G)$ such that $X = w_0(A_0 - B_0) + \cdots + w_k(A_k - B_k)$.*

Proof. Combine Corollary 22 and Lemma 23, getting

$$X = \sum_{i=0}^{k} 2^i \cdot Y_i = \sum_{i=0}^{k} 2^i \cdot (A_i - B_i)$$

where $\tilde{c}(A_i + B_i) \leq \text{mccc}(G)$. □

The proof of Theorem 24 suggests a straightforward algorithm for $\text{MCCD}_{\mathbb{Z}}$, which we detail in Algorithm 2 and describe at a high level here. First, iteratively decompose X, yielding $\log\lceil \|X\| \rceil + 1$ unitary circulations. Then use Lemma 23 to decompose each into two circulations of cost at most $\text{mccc}(G)$. However, $\text{mccc}(G)$ is not necessarily a lower bound on $\text{MCCD}_{\mathbb{Z}}$ if the circulation is 0 on some edges, and thus this approach does not directly derive an approximation. To overcome this issue, we instead find a circulation decomposition of a spanning subgraph G' of G for which $\text{mccc}(G')$ lower bounds $\text{mccd}_{\mathbb{Z}}(G, X)$. Namely, we first find a minimum cost circulation cover in G of the subset S of edges with non-zero flow in $O(n \log m(m + n \log n))$ time (according to Lemma 20),

ALGORITHM 1: UNITARY(G,X): Produces a Unitary Circulation Y from an Input Circulation X Such that $X \equiv_2 Y$, as in Lemma 21

```
E_odd ← odd edges of G, undirected
C ← a decomposition of G_odd = (V, E_odd) into cycles, oriented arbitrarily
E+_odd ← directed edges of C
for (u, v) ∈ E do
    if (u, v) ∈ E+_odd then
        Y(u, v) ← +1
    else if (v, u) ∈ E+_odd then
        Y(u, v) ← −1
    else
        Y(u, v) ← 0
    end if
end for
return Y
```

ALGORITHM 2: Finds The Circulation Decomposition of Theorem 24

```
Compute a minimum cost circulation cover of {(u, v) ∈ E | X(u, v) ≠ 0} {Lemma 20}
Remove from G any edge not covered by this circulation cover to obtain G′
𝒫 ← [], 𝒲 ← [] {length-zero vectors}
C ← circulation of cost mccc(G′), C ≥ 1 {Lemma 20}
D ← UNITARY(G′, C); if Val(D) < 0 set D = −D{Algorithm 1}
i ← 0
while ||X|| > 1 do
    Y_i ← UNITARY(G′, X){Algorithm 1}
    X ← (X − Y_i)/2
    i ← i + 1
end while
Y_i ← X
for j ∈ {0, . . . , i} s.t. Y_j ≠ 0 do
    A ← C − D + Y_j, B ← C − D − Y_j
    Concatenate A and B to 𝒫
    Concatenate 2^j and −2^j to 𝒲
end for
return (𝒫, 𝒲)
```

and then remove from G any edge not covered by the circulation, obtaining G'. By construction, the cost of this circulation cover is a lower bound of $\mathrm{mccd}_{\mathbb{Z}}(G, X)$. Moreover, the cost of this circulation cover is exactly $\mathrm{mccc}(G')$, since every circulation cover of G' is also a circulation cover of S in G.

To prove the correctness of Algorithm 2, we first define a a subroutine implementing Lemma 21.

LEMMA 25. *Algorithm 1 returns a unitary circulation from an input circulation Y such that $X \equiv_2 Y$, as in Lemma 21, in $O(m)$ time.*

PROOF. The correctness of the algorithm is given by Lemma 21. Finally, the first 3 subroutines as well as the entire `for-loop` takes $O(m)$ time. □

THEOREM 3. *$MCCD_{\mathbb{Z}}$ can be approximated with a factor of $\log\lceil\|X\|\rceil + 1$ in runtime $O(n \log m(m + n \log n) + m \log \|X\|)$.*

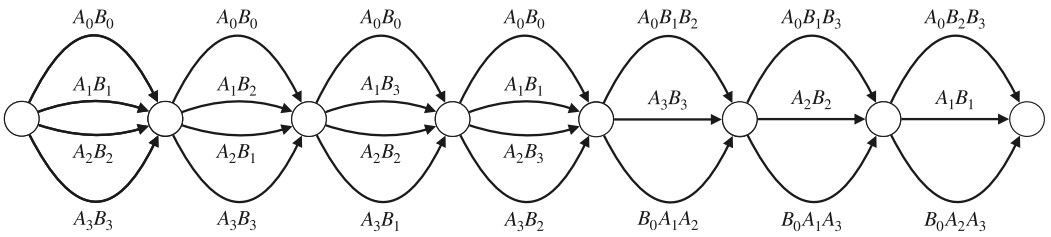


Fig. 9. Paths A_i and B_i ($i \in \{0,1,2,3\}$), each edge being labeled with the paths it appears in. Assign to each path A_i weight a_i, and to each path B_i weight b_i, such that $a_0 = b_0 = 3$, and $a_i = 6^{2i} + 1$ and $b_i = 6^{2i+1} + 5$ for $i = 1, 2, 3$. Define the flow X on G as $X = \sum_{i=0}^{3} a_i A_i + \sum_{i=0}^{3} b_i B_i$. Note that these weights are a solution of k-FWA$_{\mathbb{N}}$ on input (G, X) with given paths A_i, B_i ($i \in \{0,1,2,3\}$).

Proof. By Theorem 24 and our previous discussion, Algorithm 2 returns a circulation decomposition for X with no more cost than $(\lceil \log \|X\| \rceil + 1) \cdot \text{mccc}(G') \leq (\lceil \log \|X\| \rceil + 1) \cdot \text{mccd}_{\mathbb{Z}}(G, X)$. We analyse the runtime line by line. Lines 1 and 4 take $O(n \log m(m + n \log n))$ time by Lemma 20. The call to Algorithm 1 on line 5 takes $O(m)$ time by Lemma 25, and checking the cost of D and flipping signs (if necessary) also takes $O(m)$ time. By Corollary 22, the while loop on line 7 executes at most $\log\lceil \|X\| \rceil + 1$ times, meaning that the entire execution takes $O(m \log \|X\|)$ time since line 8 takes $O(m)$ time by Lemma 25. Since there are at most $\log\lceil \|X\| \rceil + 1$ Y_i's, the for loop on line 13 executes at most $\log\lceil \|X\| \rceil + 1$ times. Each execution of the for-loop finds two circulations of total cost of at most mccc(G') in $O(m)$ time, so the whole also loop takes $O(m \log \|X\|)$ time. Thus, the overall runtime is $O(n \log m(m + n \log n) + m \log \|X\|)$. □

With the reduction given in Lemma 18, we obtain an approximation algorithm of the same ratio for MFD$_{\mathbb{Z}}$. However, we can improve the runtime of Lemma 20:

Corollary 26. *Algorithm 2 is also a* $\log\lceil \|X\| \rceil + 1$*-approximation for MFD*$_{\mathbb{Z}}$ *with runtime* $O(m(n + \mathit{width}(G) \log\lceil \|X\| \rceil))$.

Proof. This is directly achieved by using Theorem 3 with Lemma 18 and by calculating the width(G) according to Lemma 7. Note that the flows A and B need to be trivially decomposed into at most width(G) paths, causing the additional factor in the runtime. □

A theorem analogous to Theorem 24 for MCCD$_{\mathbb{N}}$ is desirable, but cannot be achieved directly with the previous methods, as Lemma 21 makes use of negative weights. However, the approach can be adapted for MCCD$_{\mathbb{N}}$ if the input flows are width-stable (Definition 11), and if it is possible to "fix" the odd flows to be even with only mccc(G) unitary flows, which we leave as an open question.

5 SOLVING THE k-FLOW WEIGHT ASSIGNMENT PROBLEM

In this section, we consider a restriction of MFD from Kloster et al. [2018] (see Figure 9 for an example).

Definition 27 (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 $L_{\mathbb{Y}} = L_{\mathbb{Y}}(P_1, \ldots, P_k) = \{w \in \mathbb{Y}^k \mid X = \sum_{j=1}^{k} P_k w_k\}$ the linear system corresponding to the paths $P_1, \ldots, P_k$.

We shortly discuss how to solve k-FWA$_{\mathbb{Z}}$. 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, for example, [Schrijver 1986, Chapter 5].

Solving k-FWA$_{\mathbb{N}}$ turns out to be more difficult, its the linear system contains the inequality $w \geq 0$. In fact, it was shown [Kloster et al. 2018] that k-FWA$_{\mathbb{N}}$ is NP-hard. The program Toboggan [Kloster et al. 2018] implements a linear FPT algorithm for MFD$_{\mathbb{N}}$ and one step of the algorithm is to solve k-FWA$_{\mathbb{N}}$ using an ILP [Kloster et al. 2018]. The authors state the following conjecture.

Conjecture 28 (Kloster et al. [2018]). *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.*

In case of a fractional decomposition (in which the weights of the paths are allowed to be rational non-negative numbers), it is indeed true that the induced linear system is of full rank k [Vatinlen et al. 2008]. As mentioned in the introduction, if the conjecture turned out to be true for natural numbers, 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 9 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 9, thus proving that the rank of the corresponding linear system is strictly less than 8.

To disprove Conjecture 28, 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), we only explain the intuition behind the construction in Figure 9 and behind the correctness proof. However, as an additional check we also ran both Toboggan [Kloster et al. 2018] and a recently developed ILP solver for MFD$_{\mathbb{N}}$ [Dias et al. 2022] on this instance, both returning $\mathrm{mfd}_{\mathbb{N}}(G, X) = 8$.

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 4 (right part). We say that a path is **fixed** if every minimum flow decomposition of the graph contains this path. The paths A_i and B_i have exponentially growing weight for growing i and get shuffled around with different permutations of the paired labels $A_i B_j$ on the left part. Due to the exponential growth, ensuring the correct parity on all edges of the right part, we can fix 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 $A_0 B_0$. 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, B_2, and B_3 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 article 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,

we generalised MFD to the problem to minimising the cost of a circluation decompisition, and showed that the integer version can be approximated even better by combining parity arguments of unitary circulations and a decomposition of such circulations of cost equal to the minimum cost to cover the graph. Finally, we have corroborated the complexity gap between the positive integer and the full integer case by disproving a conjecture from Kloster et al. [2018] (also motivating the heuristic in Shao and Kingsford [2017]), which would have had sped up their FPT algorithm for $\text{MFD}_{\mathbb{N}}$.

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

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

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

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

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

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

Fernando H. C. Dias, Manuel Cáceres, Lucia Williams, Brendan Mumey, and Alexandru I. Tomescu, 2023. A safety framework for flow decomposition problems via integer linear programming. *Bioinformatics* 39, 11 (2023), btad640. https://doi.org/10.1093/bioinformatics/btad640

Fernando H. C. Dias, Lucia Williams, Brendan Mumey, and Alexandru I. Tomescu. 2022. Fast, flexible, and exact minimum flow decompositions via ILP. In *Proceedings of the Research in Computational Molecular Biology: 26th Annual International Conference*. Springer, 230–245.

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

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

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

Harold N. Gabow and Robert E. Tarjan. 1989. Faster scaling algorithms for network problems. *SIAM Journal on Computing* 18, 5 (1989), 1013–1036.

Marcelo Garlet Millani, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. 2020. Efficient algorithms for measuring the funnel-likeness of DAGs. *Journal of Combinatorial Optimization* 39, 1 (2020), 216–245.

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

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

Amit Jain and Naga Chandrasekharan. 1993. An efficient parallel algorithm for min-cost flow on directed series-parallel networks. In *Proceedings of the 7th International Parallel Processing Symposium*. 188–192. DOI: https://doi.org/10.1109/IPPS.1993.262879

Shahbaz Khan, Milla Kortelainen, Manuel Cáceres, Lucia Williams, and Alexandru I. Tomescu. 2022. Improving rna assembly via safety and completeness in flow decompositions. *Journal of Computational Biology* 29, 12 (2022), 1270–1287.

Kyle Kloster, Philipp Kuinke, Michael P. O.'Brien, Felix Reidl, Fernando Sánchez Villaamil, Blair D. Sullivan, and Andrew van der Poel. 2018. A practical fpt algorithm for flow decomposition and transcript assembly. In *2018 Proceedings of the 20th Workshop on Algorithm Engineering and Experiments*. SIAM, 75–86.

Brendan Mumey, Samareh Shahmohammadi, Kathryn McManus, and Sean Yaw. 2015. Parity balancing path flow decomposition and routing. In *Proceedings of the 2015 IEEE Globecom Workshops*. IEEE, 1–6.

Nils Olsen, Natalia Kliewer, and Lena Wolbeck. 2022. 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* 30, 3 (2022), 883–919. DOI: https://doi.org/10.1007/s10100-020-00705-6

James B. Orlin. 2013. Max flows in O(nm) time, or better. In *Proceedings of the Symposium on Theory of Computing Conference,, Palo Alto, CA, USA, June 1–4, 2013*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM, 765–774. DOI: https://doi.org/10.1145/2488608.2488705

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

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

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

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. 2016. A benchmark for RNA-seq quantification pipelines. *Genome Biology* 17, 1 (2016), 1–12.

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

Alexandru I. Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. 2013. A novel min-cost flow method for estimating transcript expression with RNA-Seq. *BMC bioinformatics* 14, 5 (2013), S15:1–S15:10.

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

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

Lucia Williams, Gillian Reynolds, and Brendan Mumey. 2019. RNA transcript assembly using inexact flows. In *Proceedings of the 2019 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE, 1907–1914.

Received 29 April 2023; revised 23 December 2023; accepted 14 January 2024