# Finding Diverse Minimum s-t Cuts

**Mark de Berg**
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

**Andrés López Martínez**
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

**Frits Spieksma**
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

## Abstract

Recently, many studies have been devoted to finding *diverse* solutions in classical combinatorial problems, such as Vertex Cover (Baste et al., IJCAI'20), Matching (Fomin et al., ISAAC'20) and Spanning Tree (Hanaka et al., AAAI'21). Finding diverse solutions is important in settings where the user is not able to specify all criteria of the desired solution. Motivated by an application in the field of *system identification*, we initiate the algorithmic study of $k$-Diverse Minimum s-t Cuts which, given a directed graph $G = (V, E)$, two specified vertices $s, t \in V$, and an integer $k > 0$, asks for a collection of $k$ minimum $s$-$t$ cuts in $G$ that has maximum diversity. We investigate the complexity of the problem for two diversity measures for a collection of cuts: (i) the sum of all pairwise Hamming distances, and (ii) the cardinality of the union of cuts in the collection. We prove that $k$-Diverse Minimum s-t Cuts can be solved in strongly polynomial time for both diversity measures via *submodular function minimization*. We obtain this result by establishing a connection between ordered collections of minimum $s$-$t$ cuts and the theory of distributive lattices. When restricted to finding only collections of mutually disjoint solutions, we provide a more practical algorithm that finds a maximum set of pairwise disjoint minimum $s$-$t$ cuts. For graphs with small minimum $s$-$t$ cut, it runs in the time of a single *max-flow* computation. These results stand in contrast to the problem of finding $k$ diverse *global* minimum cuts – which is known to be NP-hard even for the disjoint case (Hanaka et al., AAAI'23) – and partially answer a long-standing open question of Wagner (Networks 1990) about improving the complexity of finding disjoint collections of minimum $s$-$t$ cuts.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** S-T MinCut, Diversity, Lattice Theory, Submodular Function Minimization

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2023.24

**Related Version** *Full Version*: `https://arxiv.org/abs/2303.07290` [6]

**Funding** This research was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 945045, and by the NWO Gravitation project NETWORKS under grant no. 024.002.003.

**Acknowledgements** We thank Martin Frohn for bringing the theory of lattices to our attention, and for fruitful discussions on different stages of this work.

## 1 Introduction

The Minimum $s$-$t$ Cut problem is a classic combinatorial optimization problem. Given a directed graph $G = (V, E)$ and two special vertices $s, t \in V$, the problem asks for a subset $S \subseteq E$ of minimum cardinality that separates vertices $s$ and $t$, meaning that removing these edges from $G$ ensures there is no path from $s$ to $t$. Such a set is called a *minimum $s$-$t$ cut* or *$s$-$t$ mincut*, and it need not be unique. This problem has been studied extensively and has numerous practical and theoretical applications. Moreover, it is known to be solvable in polynomial time. Several variants and generalizations of the problem have been studied;


we mention the *global minimum cut* problem and the problem of *enumerating all minimum s-t cuts* in a graph. In this paper, we initiate the algorithmic study of computing *diverse* minimum *s-t* cuts. Concretely, we introduce the following optimization problem.

**$k$-Diverse Minimum s-t Cuts ($k$-DMC).** *Given are a directed graph $G = (V, E)$, vertices $s, t \in V$, and an integer $k > 0$. Let $\Gamma_G(s,t)$ be the set of minimum s-t cuts in $G$, and let $U_k$ be the set of $k$-element multisets of $\Gamma_G(s,t)$. We want to find $C \in U_k$ such that $d(C) = \max_{S \in U_k} d(S)$, where $d : U_k \to \mathbb{N}$ is a measure of diversity.*

Informally, given a directed graph $G$, vertices $s$ and $t$, and an integer $k$, we are interested in finding a collection of $k$ *s-t* mincuts in $G$ that are as different from each other as possible; that is, a collection having *maximum diversity*. Finding diverse solution sets is important in settings where the user is not able to specify all criteria of the desired solution. We mention the *synthesis problem* as an application of diverse minimum *s-t* cuts [27, 28].

To formally capture the notion of diversity of a collection of sets, several measures have been proposed (e.g., [30, 2, 15, 1, 13]). In this work, we choose two natural and general measures as our notions of diversity. Given a collection $(X_1, X_2, \ldots, X_k)$ of subsets of a set $A$ (not necessarily distinct), we define $d_{\text{sum}}(X_1, \ldots, X_k) = \sum_{1 \le i < j \le k} |X_i \triangle X_j|$ and $d_{\text{cov}}(X_1, \ldots, X_k) = \left|\bigcup_{1 \le i \le k} X_i\right|$, where $X_i \triangle X_j = (X_i \cup X_j) \setminus (X_i \cap X_j)$ is the *symmetric difference* (or *Hamming distance*) of $X_i$ and $X_j$. We call $d_{\text{sum}}$ and $d_{\text{cov}}$ the *pairwise-sum* and *coverage* diversity measures, respectively.

**Our results.** We investigate the complexity of the following two variants of $k$-Diverse Minimum s-t Cuts: (i) Sum $k$-Diverse Minimum s-t Cuts (Sum-$k$-DMC), and (ii) Cover $k$-Diverse Minimum s-t Cuts (Cov-$k$-DMC). These are the problems obtained when defining function $d$ in $k$-DMC as diversity measures $d_{\text{sum}}$ and $d_{\text{cov}}$, respectively. For a graph $G$, we use $n$ to denote the number of nodes and $m$ to denote the number of edges.

Contrary to the hardness of finding diverse *global* mincuts in a graph [13], we show that both Sum-$k$-DMC and Cov-$k$-DMC can be solved in polynomial time. We show this via a reduction to the *submodular function minimization* problem (SFM) on a *lattice*, which is known to be solvable in strongly polynomial time when the lattice is *distributive* [10, 16, 26].

▶ **Theorem 1.** *Sum-$k$-DMC and Cov-$k$-DMC can be solved in strongly polynomial time.*

At the core of this reduction is a generalization of an old result establishing a connection between minimum *s-t* cuts and distributive lattices [7]. As will be elaborated in Section 3, we obtain our results by showing that the pairwise-sum and coverage diversity measures (reformulated as minimization objectives) are *submodular* functions on the lattice $L^*$ defined by *left-right ordered* collections of *s-t* mincuts and that this lattice is in fact *distributive*. Using the current fastest algorithm for SFM [17], together with an appropriate representation of the lattice $L^*$, we can obtain an algorithm that solves these problems in $O(k^5 n^5)$ time.

In Section 4, we obtain better time bounds for the special case of finding collections of *s-t* mincuts that are pairwise disjoint. Similar to SUM-$k$-DMC and COV-$k$-DMC, our approach exploits the partial order structure of *s-t* mincuts. We use this to efficiently solve the following optimization problem, which we call $k$-Disjoint Minimum *s-t* Cuts: given a graph $G = (V, E)$, vertices $s, t \in V$, and an integer $k \le k_{\max}$, find $k$ pairwise disjoint *s-t* mincuts in $G$. Here, $k_{\max}$ denotes the maximum number of disjoint *s-t* mincuts in $G$. Our algorithm is significantly simpler than the previous best algorithm by Wagner [29], which runs in the time of a poly-logarithmic number of calls to any *min-cost flow* algorithm. Our algorithm takes $O(F(m,n) + m\lambda)$ time, where $F(m,n)$ is the time required by a unit-capacity

*max-flow* computation, and $\lambda$ is the size of an $s$-$t$ mincut in the graph. By plugging in the running time of the current fastest deterministic max-flow algorithms [21, 18], we obtain the following time bounds. When $\lambda \leq m^{1/3+o(1)}$, our algorithm improves upon the previous best runtime for this problem.

▶ **Theorem 2.** *$k$-Disjoint Minimum $s$-$t$ Cuts can be solved in time $O(m^{4/3+o(1)} + m\lambda)$.*

**Related Work.** Many efforts have been devoted to finding diverse solutions in combinatorial problems. In their seminal paper [20], Kuo *et al.* were the first to explore this problem from a complexity-theoretic perspective. They showed that the basic problem of maximizing a distance norm over a set of elements is already NP-hard. Since then, the computational complexity of finding diverse solutions in many other combinatorial problems has been studied. For instance, diverse variants of Vertex Cover, Matching and Hitting Set have been shown to be NP-hard, even when considering simple diversity measures like the pairwise-sum of Hamming distances, or the minimum Hamming distance between sets. This has motivated the study of these and similar problems from the perspective of *fixed-parameter tractable* (FPT) algorithms [1, 2, 8]. Along the same line, Hanaka et al. [13] recently developed a framework to design approximation algorithms for diverse variants of combinatorial problems. On the positive side, diverse variants of other classic problems are known to be polynomially solvable, such as Spanning Tree [15], Shortest Path [14, 30], and Bipartite Matching [14], but not much is known about graph partitioning problems in light of diversity.

The problem of finding multiple minimum cuts has received considerable attention [13, 25, 29]. Picard and Queyranne [25] initiated the study of finding all minimum $s$-$t$ cuts in a graph, showing that these can be enumerated efficiently. They observe that the closures of a naturally-defined poset over the vertices of the graph, correspond bijectively to minimum $s$-$t$ cuts. An earlier work of Escalante [7] already introduced an equivalent poset for minimum $s$-$t$ cuts, but contrary to Picard and Queyranne, no algorithmic implications were given. Nonetheless, Escalante shows that the set of $s$-$t$ mincuts in a graph, together with this poset, defines a distributive lattice. Similar structural results for *stable matchings* and *circulations* have been shown to have algorithmic implications [11, 19], but as far as we know, the lattice structure of $s$-$t$ mincuts has been seldomly exploited in the algorithmic literature.

Wagner [29] studied the problem of finding $k$ pairwise-disjoint $s$-$t$ cuts of *minimum total cost* in an edge-weighted graph. He showed that this problem can be solved in polynomial time by means of a reduction to a *transshipment* problem; where he raised the question of whether improved complexity bounds were possible by further exploiting the structure of the problem, as opposed to using a general purpose *min-cost flow* algorithm for solving the transshipment formulation. In sharp contrast, Hanaka et al. [13] recently established that the problem of finding $k$ pairwise-disjoint *global* minimum cuts in a graph is NP-hard (for $k$ part of the input). We are not aware of any algorithm for minimum $s$-$t$ cuts that runs in polynomial time with theoretical guarantees on diversity.

## 2 Preliminaries

### 2.1 Distributive Lattices

In this paper, we use properties of distributive lattices. Here we introduce some basic concepts and results. For a more detailed introduction to lattice theory see e.g., [3, 5, 9].

A *partially ordered set (poset)* $P = (X, \preceq)$ is a ground set $X$ together with a binary relation $\preceq$ on $X$ that is reflexive, antisymmetric, and transitive. For a poset $P = (X, \preceq)$, an *ideal* is a set $U \subseteq X$ where $u \in U$ implies that $v \in U$ for all $v \preceq u$. We use $\mathcal{D}(P)$ to denote

the family of all ideals of $P$. When the binary operation $\preceq$ is clear from the context, we use the same notation for a poset and its ground set. We consider the standard representation of a poset $P$ as a directed graph $G(P)$ containing a node for each element and edges from an element to its predecessors. In terms of $G(P) = (V, E)$, a subset $W$ of $V$ is an ideal if and only if there is no outgoing edge from $W$.

A *lattice* is a poset $L = (X, \preceq)$ in which any two elements $x, y \in X$ have a (unique) greatest lower bound, or *meet*, denoted by $x \wedge y$, as well as a (unique) least upper bound, or *join*, denoted by $x \vee y$. Hence, a lattice can also be identified by the tuple $(X, \vee, \wedge)$. A lattice $L'$ is a *sublattice* of $L$ if $L' \subseteq L$ and $L'$ has the same meet and join operations as $L$. In this paper, we only consider *distributive lattices*, which are lattices whose meet and join operations satisfy distributivity; that is, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ and $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, for any $x, y, z \in L$. Note that a sublattice of a distributive lattice is also distributive.

Suppose we have a collection $L_1, \ldots, L_k$ of lattices $L_i = (X_i, \vee_i, \wedge_i)$ with $i \in \{1, ..., k\}$. The *(direct) product lattice* $L_1 \times \cdots \times L_k$ is a lattice with ground set $X = \{(x_1, \ldots, x_k) : x_i \in X_i\}$ and join $\vee$ and meet $\wedge$ acting component-wise; that is, $x \vee y = (x_1 \vee_1 y_1, \ldots, x_k \vee_k y_k)$ and $x \wedge y = (x_1 \wedge_1 y_1, \ldots, x_k \wedge_k y_k)$ for any $x, y \in X$. The lattice $L^k$ is the product lattice of $k$ copies of $L$ and is called the $k$th *power* of $L$. If $L$ is distributive, then $L^k$ is also distributive.

A crucial notion in this work is that of *join-irreducibles*. An element $x$ of a lattice $L$ is called *join-irreducible* if it cannot be expressed as the join of two elements $y, z \in L$ with $y, z \neq x$. In a lattice, any element is equal to the join of all join-irreducible elements lower than or equal to it. The set of join-irreducible elements of $L$ is denoted by $J(L)$. Note that $J(L)$ is a poset whose order is inherited from $L$. Due to Birkhoff's representation theorem every distributive lattice $L$ is isomorphic to the lattice $\mathcal{D}(J(L))$ of ideals of its poset of join-irreducibles, with union and intersection as join and meet operations. Hence, a distributive lattice $L$ can be uniquely recovered from its poset $J(L)$.

▶ **Theorem 3** (Birkhoff's Representation Theorem [3]). *Any distributive lattice $L$ can be represented as the poset of its join-irreducibles $J(L)$, with the order induced from $L$.*

For a distributive lattice $L$, this implies that there is a *compact representation* of $L$ as the directed graph $G(L)$ that characterizes its set of join-irreducibles. (The graph $G(L)$ is unique if we remove transitive edges.) This is useful when designing algorithms, as the size of $G(L)$ is $O(|J(L)|^2)$, while $L$ can have as many as $2^{|J(L)|}$ elements.

## 2.2 Submodular Function Minimization

Let $f$ be a real-valued function on a lattice $L = (X, \preceq)$. We say that $f$ is *submodular* on $L$ if $f(x \wedge y) + f(x \vee y) \leq f(x) + f(y)$, for all $x, y \in X$. If $-f$ is submodular in $L$, then we say that $f$ is *supermodular* in $L$ and just *modular* if $f$ is both sub and supermodular. The *submodular function minimization* problem (SFM) on lattices is, given a submodular function $f$ on $L$, to find an element $x \in L$ such that $f(x)$ is minimum. An important fact that we use in our work is that the sum of submodular functions is also submodular. Also, note that minimizing $f$ is equivalent to maximizing $-f$.

Consider the special case of a lattice whose ground set $X \subseteq 2^U$ is a family of subsets of a set $U$, and meet and join are intersection and union of sets, respectively. It is known that any submodular function $f$ on such a lattice can be minimized in polynomial time in $|U|$ [10, 16, 26], assuming that for any $Y \subseteq U$, the value of $f(Y)$ is given by an *evaluation oracle* that also runs in polynomial time in $|U|$. The current fastest algorithm for SFM on sets runs in $O(|U|^3 T_{\mathrm{EO}})$ time [17], where $T_{\mathrm{EO}}$ is the time required for one call to the evaluation oracle.

Due to Birkhoff's theorem, the seemingly more general case of SFM on distributive lattices can be reduced to SFM on sets (see [4, Sec. 3.1] for details). Hence, any polynomial-time algorithm for SFM on sets can be used to minimize a submodular function $f$ defined on a distributive lattice $L$. An important remark is that the running time now depends on the size of the set $J(L)$ of join-irreducibles.

▶ **Theorem 4** ([24, Note 10.15] & [22, Thm.1]). *For any distributive lattice $L$, given by its poset of join-irreducibles $J(L)$, a submodular function $f : L \to \mathbb{R}$ can be minimized in polynomial time in $|J(L)|$, provided a polynomial time evaluation oracle for $f$.*

## 2.3 Minimum Cuts

Throughout this paper, we restrict our discussion to directed graphs. All our results can be extended to undirected graphs by means of well-known transformations. Likewise, we deal only with edge-cuts, although our approach can be easily adapted to vertex-cuts as well.

Let $G$ be a directed graph with vertex set $V(G)$ and edge set $E(G)$. As usual, we define $n \coloneqq |V(G)|$ and $m \coloneqq |E(G)|$. Given a *source* $s \in V(G)$ and *target* $t \in V(G)$ in $G$, we call a subset $X \subset E(G)$ an *$s$-$t$ cut* if the removal of $X$ from the graph ensures that no path from $s$ to $t$ exists in $G \setminus X$. The size of a cut is the total number of edges it contains. If an $s$-$t$ cut in $G$ has smallest size $\lambda(G)$, we call it a minimum $s$-$t$ cut, or an $s$-$t$ mincut. Note that such a cut need not be unique (in fact, there can be exponentially many). To denote the set of all $s$-$t$ mincuts of $G$, we use $\Gamma_G(s,t)$.

A (directed) path starting in a vertex $u$ and ending in a vertex $v$ is called a $u$-$v$ path. By Menger's theorem, the cardinality of a minimum $s$-$t$ cut in $G$ is equal to the maximum number of internally edge-disjoint $s$-$t$ paths in the graph. Let $\mathcal{P}_{s,t}(G)$ denote a maximum-sized set of edge-disjoint paths from $s$ to $t$ in $G$. Note that any minimum $s$-$t$ cut in $G$ contains exactly one edge from each path in $\mathcal{P}_{s,t}(G)$. For two distinct edges (resp. vertices) $x$ and $y$ in a $u$-$v$ path $p$, we say that $x$ is a path-predecessor of $y$ in $p$ and write $x \prec_p y$ if the path $p$ meets $x$ before $y$. We use this notation indistinctly for edges and vertices. It is easily seen that the relation $\prec_p$ extends uniquely to a non-strict partial order. We denote this partial order by $x \preceq_p y$. Consider now any subset $W \subseteq \Gamma_G(s,t)$ of $s$-$t$ mincuts in $G$. We let $E(W) = \bigcup_{X \in W} X$. Two crucial notions in this work are those of *leftmost* and *rightmost* $s$-$t$ mincuts. The *leftmost* $s$-$t$ mincut in $W$ consists of the set of edges $S_{\min}(W) \subseteq E(W)$ such that, for every path $p \in \mathcal{P}(s,t)$, there is no edge $e \in E(W)$ satisfying $e \prec_p f$ for any $f \in S_{\min}(W)$. Similarly for the *rightmost* $s$-$t$ mincut $S_{\max}(W) \subseteq E(W)$. Note that both $S_{\min}(W)$ and $S_{\max}(W)$ are also $s$-$t$ mincuts in $G$ (see the proof of Proposition 5 in the full version of the paper [6]). When $W$ consists of the entire set of $s$-$t$ mincuts in $G$, we denote these extremal cuts by $S_{\min}(G)$ and $S_{\max}(G)$.

On the set of $s$-$t$ cuts (not necessarily minimum), the following predecessor-successor relation defines a partial order: an $s$-$t$ cut $X$ is a predecessor of another $s$-$t$ cut $Y$, denoted by $X \leq Y$, if every path from $s$ to $t$ in $G$ meets an edge of $X$ at or before an edge of $Y$. The set of $s$-$t$ mincuts together with relation $\leq$ defines a distributive lattice $L$ [7, 23]. Moreover, a compact representation $G(L)$ can be constructed from a maximum flow in linear time [25]. These two facts play a crucial role in the proof of our main result in the next section.

# 3 A Polynomial Time Algorithm for SUM-k-DMC and COV-k-DMC

This section is devoted to proving Theorem 1 by reducing SUM-$k$-DMC and COV-$k$-DMC to SMF on distributive lattices. First, we show that the domain of solutions of SUM-$k$-DMC and COV-$k$-DMC can be restricted to the set of $k$-tuples that satisfy a particular order, as opposed to the set of $k$-sized multisets of $s$-$t$ mincuts (see Corollary 6 below). The reason

for doing so is that the structure provided by the former set can be exploited to assess the "modularity" of the total-sum and coverage objectives. Next, we introduce the notions of *left-right order* and *edge multiplicity*, which are needed throughout the section.

Consider a graph $G$ with specified $s, t \in V(G)$, and let $U^k$ be the set of all $k$-tuples over $\Gamma_G(s,t)$. An element $C \in U^k$ is a (ordered) collection or sequence $[X_1, \ldots, X_k]$ of cuts $X_i \in \Gamma_G(s,t)$, where $i$ runs over the index set $\{1, \ldots, k\}$. We say that $C$ is in *left-right order* if $X_i \leq X_j$ for all $i < j$. Let us denote by $U^k_{\text{lr}} \subseteq U^k$ the set of all $k$-tuples over $\Gamma_G(s,t)$ that are in left-right order. Then, for any two $C_1, C_2 \in U^k_{\text{lr}}$, with $C_1 = [X_1, \ldots, X_k]$, $C_2 = [Y_1, \ldots, Y_k]$, we say that $C_1$ is a *predecessor* of $C_2$ (and $C_2$ a *successor* of $C_1$) if $X_i \leq Y_i$ for all $i \in [k]$, and denote this by $C_1 \preceq C_2$. Now, consider again a collection $C \in U^k$. The set of edges $\bigcup_{X \in C} X$ is denoted by $E(C)$. We define the *multiplicity* of an edge $e \in E(G)$ with respect to (w.r.t.) $C$ as the number of cuts in $C$ that contain $e$ and denote it by $\mu_e(C)$. We say that an edge $e \in E(C)$ is a *shared edge* if $\mu_e(C) \geq 2$. The set of shared edges in $C$ is denoted by $E_{\text{shr}}(C)$. We make the following proposition, the proof of which is in the full version of the paper [6].

▶ **Proposition 5.** *For every $C \in U^k$ there exists $\hat{C} \in U^k_{\text{lr}}$ such that $\mu_e(C) = \mu_e(\hat{C})$ $\forall e \in E(G)$.*

In other words, given a $k$-tuple of $s$-$t$ mincuts, there always exists a $k$-tuple on the same set of edges that is in left-right order; each edge occurring with the same multiplicity. Consider now the total-sum and the coverage diversity measures first introduced in Section 1. We can rewrite them directly in terms of the multiplicity of shared edges as

$$d_{\text{sum}}(C) = 2\left[\lambda(G)\binom{k}{2} - \sum\nolimits_{e \in E_{\text{shr}}(C)} \binom{\mu_e(C)}{2}\right] \quad \text{and} \tag{1}$$

$$d_{\text{cov}}(C) = k\lambda(G) - \sum\nolimits_{e \in E_{\text{shr}}(C)} (\mu_e(C) - 1)\,, \tag{2}$$

where terms outside the summations are constant terms. Then, combining eq. (1) (resp. (2)) with Proposition 5, we obtain the following corollary. (For simplicity, we state this only for the $d_{\text{sum}}$ diversity measure, but an analogous claim holds for the $d_{\text{cov}}$ measure.)

▶ **Corollary 6.** *Let $C \in U^k$ such that $d_{\text{sum}}(C) = \max_{S \in U^k} d_{\text{sum}}(S)$. Then there exists $C' \in U^k_{\text{lr}}$ such that $d_{\text{sum}}(C') = d_{\text{sum}}(C)$.*

This corollary tells us that in order to solve SUM-$k$-DMC (resp. COV-$k$-DMC) we do not need to optimize over the set $U_k$ of $k$-element multisets of $\Gamma_G(s,t)$. Instead, we can look at the set $U^k_{\text{lr}} \subseteq U^k$ of $k$-tuples that are in left-right order. Moreover, it follows from Eqs. (1) and (2) that the problem of maximizing $d_{\text{sum}}(C)$ and $d_{\text{cov}}(C)$ is equivalent to minimizing

$$\hat{d}_{\text{sum}}(C) = \sum\nolimits_{e \in E_{\text{shr}}(C)} \binom{\mu_e(C)}{2}, \quad \text{and} \tag{3}$$

$$\hat{d}_{\text{cov}}(C) = \sum\nolimits_{e \in E_{\text{shr}}(C)} (\mu_e(C) - 1)\,, \tag{4}$$

respectively. In turn, the submodularity of $\hat{d}_{\text{sum}}(C)$ (resp. $\hat{d}_{\text{cov}}(C)$) implies the supermodularity of $d_{\text{sum}}(C)$ (resp. $d_{\text{cov}}(C)$) and vice versa. In the remaining of the section, we shall only focus on the minimization objectives $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$.

We are now ready to show that both SUM-$k$-DMC and COV-$k$-DMC can be reduced to SFM. We first show that the poset $L^* = (U^k_{\text{lr}}, \preceq)$ is a distributive lattice (Section 3.1). Next we prove that the diversity measures $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$ are submodular functions on $L^*$ (Section 3.2). Lastly, we show that there is a compact representation of the lattice $L^*$ and that it can be constructed in polynomial time, concluding with the proof of Theorem 1 (Section 3.3).

## 3.1 Proof of Distributivity

We use the following result of Escalante [7] (see also [12] or [23, Thm. 4]). Recall that $\leq$ denotes the predecessor-successor relation between two $s$-$t$ mincuts.

▶ **Lemma 7.** *The set $\Gamma_G(s,t)$ of $s$-$t$ mincuts of $G$ together with the binary relation $\leq$ forms a distributive lattice $L$. For any two cuts $X, Y \in L$, the join and meet operations are given by $X \vee Y := S_{\max}(X \cup Y)$, and $X \wedge Y := S_{\min}(X \cup Y)$, respectively.*

By the definition of product lattice, we can extend this result to the relation $\preceq$ on the set $U^k_{\text{lr}}$ of $k$-tuples of $s$-$t$ mincuts that are in left-right order.

▶ **Lemma 8.** *The set $U^k_{\text{lr}}$, together with relation $\preceq$, defines a distributive lattice $L^*$. For any two elements $C_1 = [X_1, \ldots, X_k]$ and $C_2 = [Y_1, \ldots, Y_k]$ in $L^*$, the join and meet operations are given by $C_1 \vee C_2 = [S_{\max}(X_1 \cup Y_1), \ldots, S_{\max}(X_k \cup Y_k)]$ and $C_1 \wedge C_2 = [S_{\min}(X_1 \cup Y_1), \ldots, S_{\min}(X_k \cup Y_k)]$, respectively.*

**Proof.** This follows directly from Lemma 7 and the definition of product lattice (see Section 2.1). Let $L^k = (U^k, \preceq)$ be the $k$th power of the lattice $L = (\Gamma_G(s,t), \leq)$ of minimum $s$-$t$ cuts, and let $L^* = (U^k_{\text{lr}}, \preceq)$ with $U^k_{\text{lr}} \subseteq U^k$ be the sublattice of left-right ordered $k$-tuples of minimum $s$-$t$ cuts. We know from Section 2 that since $L$ is distributive, then so is the power lattice $L^k$. Moreover, any sublattice of a distributive lattice is also distributive. Hence, it follows that the lattice $L^*$ is also distributive. ◀

## 3.2 Proof of Submodularity

Now we prove that the functions $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$ are submodular on the lattice $L^*$. We start with two lemmas that establish useful properties of the multiplicity function $\mu_e(C)$ on $L^*$. Due to space constraints, we defer the proofs to the full version of the paper [6].

▶ **Lemma 9.** *The multiplicity function $\mu_e : U^k_{\text{lr}} \to \mathbb{N}$ is modular on $L^*$.*

▶ **Lemma 10.** *For any two $C_1, C_2 \in L^*$ and $e \in E(C_1) \cup E(C_2)$, it holds that $\max(\mu_e(C_1 \vee C_2), \mu_e(C_1 \wedge C_2)) \leq \max(\mu_e(C_1), \mu_e(C_2))$.*

Lemma 10 plays an important role in the submodularity of $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$. In contrast to Lemma 9, it does not hold on the $k$th power lattice of the distributive lattice $L$ of Lemma 7.

**Submodularity of $\hat{d}_{\text{sum}}$.** Recall the definition of $\hat{d}_{\text{sum}}(C)$ in equation (3), and let $B_e : U^k_{\text{lr}} \to \mathbb{N}$ be the function defined by $B_e(C) = \binom{\mu_e(C)}{2}$. We can rewrite equation (3) as $\hat{d}_{\text{sum}}(C) = \sum_{e \in E_{\text{shr}}(C)} B_e(C)$. The following is an immediate consequence of Lemmas 9–10 and the convexity of $B_e(C)$.

▷ Claim 11. For any two $C_1, C_2 \in L^*$ and $e \in E(G)$, we have $B_e(C_1 \vee C_2) + B_e(C_1 \wedge C_2) \leq B_e(C_1) + B_e(C_2)$.

In other words, the function $B_e(C)$ is submodular in the lattice $L^*$. Now, recall that the sum of submodular functions is also submodular. Then, taking the sum of $B_e(C)$ over all edges $e \in E(G)$ results in a submodular function. From here, notice that $B_e(C) = 0$ for unshared edges; that is, when $\mu_e(C) < 2$. This means that such edges do not contribute to the sum. It follows that, for any two $C_1, C_2 \in L^*$, we have

$$\sum_{e \in E_{\text{shr}}(C_1 \vee C_2)} B_e(C_1 \vee C_2) + \sum_{e \in E_{\text{shr}}(C_1 \wedge C_2)} B_e(C_1 \wedge C_2) \leq \sum_{e \in E_{\text{shr}}(C_1)} B_e(C_1) + \sum_{e \in E_{\text{shr}}(C_2)} B_e(C_2).$$

Observe that each sum in the inequality corresponds to the definition of $\hat{d}_{sum}$ applied to the arguments $C_1 \vee C_2$, $C_1 \wedge C_2$, $C_1$ and $C_2$, respectively. Hence, by definition of submodularity, we obtain our desired result.

▶ **Theorem 12.** *The function $\hat{d}_{\text{sum}} : U_{\text{lr}}^k \to \mathbb{N}$ is submodular on the lattice $L^*$.*

**Submodularity of $\hat{d}_{\text{cov}}$.** Consider the function $F_e(C) : U_{\text{lr}}^k \to \mathbb{N}$ defined by $F_e(C) = \mu_e(C) - 1$. It is an immediate corollary of Lemma 9 that $F_e(C)$ is modular in $L^*$. Then, the sum $\sum_e F_e(C)$ taken over all edges $e \in E(G)$ is also modular. Notice that only shared edges in $C$ contribute positively to the sum. The contribution of unshared edges is either neutral or negative. We can split this sum into two parts: the sum over shared edges $e \in E_{\text{shr}}(C)$, and the sum over $e \in E(G) \setminus E_{\text{shr}}(C)$. The latter can be further simplified to $|E(C)| - |E(G)|$ by observing that only the edges $e \in E(G) \setminus E(C)$ make a (negative) contribution. Therefore, we write

$$\sum\nolimits_{e \in E(G)} F_e(C) = \left( \sum\nolimits_{e \in E_{\text{shr}}(C)} (\mu_e(C) - 1) \right) + |E(C)| - |E(G)|. \tag{5}$$

We know $\sum_e F_e(C)$ to be a modular function on $L^*$, hence for any two $C_1, C_2 \in L^*$ we have $\sum_{e \in E(G)} F_e(C_1 \vee C_2) + \sum_{e \in E(G)} F_e(C_1 \wedge C_2) = \sum_{e \in E(G)} F_e(C_1) + \sum_{e \in E(G)} F_e(C_2)$, which, by equation (5), is equivalent to

$$\begin{aligned} &\left( \sum_{e \in E_{\text{shr}}(C_1 \vee C_2)} (\mu_e(C_1 \vee C_2) - 1) + \sum_{e \in E_{\text{shr}}(C_1 \wedge C_2)} (\mu_e(C_1 \wedge C_2) - 1) \right) \\ &\qquad + |E(C_1 \vee C_2)| + |E(C_1 \wedge C_2)| = \\ &= \left( \sum_{e \in E_{\text{shr}}(C_1)} (\mu_e(C_1) - 1) + \sum_{e \in E_{\text{shr}}(C_2)} (\mu_e(C_2) - 1) \right) + |E(C_1)| + |E(C_2)|. \end{aligned} \tag{6}$$

Now, from Lemmas 9 and 10, we have the following result, whose proof can be found in the full version [6].

▷ Claim 13. For any two $C_1, C_2 \in L^*$ we have $|E(C_1 \vee C_2)| + |E(C_1 \wedge C_2)| \geq |E(C_1)| + |E(C_2)|$.

Given Claim 13, it is clear that to satisfy equality in equation (6) it must be that:

$$\begin{aligned} &\sum_{e \in E_{\text{shr}}(C_1 \vee C_2)} (\mu_e(C_1 \vee C_2) - 1) + \sum_{e \in E_{\text{shr}}(C_1 \wedge C_2)} (\mu_e(C_1 \wedge C_2) - 1) \\ &\qquad \leq \sum_{e \in E_{\text{shr}}(C_1)} (\mu_e(C_1) - 1) + \sum_{e \in E_{\text{shr}}(C_2)} (\mu_e(C_2) - 1), \end{aligned}$$

from which the submodularity of $\hat{d}_{\text{cov}}$ immediately follows.

▶ **Theorem 14.** *The function $\hat{d}_{\text{cov}} : U_{\text{lr}}^k \to \mathbb{N}$ is submodular on the lattice $L^*$.*

## 3.3 Finding the Set of Join-Irreducibles

We now turn to the final part of the reduction to SFM. By Lemma 8, we know that the lattice $L^*$ of left-right ordered collections of $s$-$t$ mincuts is distributive. Moreover, it follows from Theorems 12 and 14 that the objective functions $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$ are submodular in $L^*$. As discussed in Section 2.2, it now suffices to find an appropriate (compact) representation of $L^*$ in the form of its poset of join-irreducibles $J(L^*)$.

Recall the distributive lattice $L$ of $s$-$t$ mincuts of a graph $G$ defined in Lemma 7. The leftmost cut $S_{\min}(G)$ can be seen as the meet of all elements in $L$. In standard lattice notation, this smallest element is often denoted by $0_L := \bigvee_{x \in L} x$. We use the following result of Picard and Queyranne.

▶ **Lemma 15** ([25]). *Let $L$ be the distributive lattice of $s$-$t$ mincuts in a graph $G$, there is a compact representation $G(L)$ of $L$ with the following properties:*

1. *The vertex set is $J(L) \cup 0_L$,*
2. *$|G(L)| \leq |V(G)|$,*
3. *Given $G$ as input, $G(L)$ can be constructed in $F(n,m) + O(m)$ time.*

In other words, the set $J(L)$ is of size $O(n)$ and can be recovered from $G$ in the time of a single max-flow computation. Moreover, each element of $J(L)$ corresponds to an $s$-$t$ mincut in $G$. From this lemma, we obtain the following result for the poset $J(L^*)$, the proof of which is in the full version [6].

▶ **Lemma 16.** *The set of join-irreducibles of $L^*$ is of size $O(kn)$ and is given by*

$$J(L^*) = \textstyle\bigcup_{i=1}^{k} J_i, \text{ where } J_i := \{(\underbrace{0_L, \ldots, 0_L}_{i-1 \text{ times}}, \underbrace{p, \ldots, p}_{k-i+1 \text{ times}}) : p \in J(L)\}.$$

Given Lemma 16, a compact representation of the lattice $L^*$ can be obtained as the directed graph $G(L^*)$ that characterizes its poset of join-irreducibles $J(L^*)$ in polynomial time (since $|J(L^*)|$ is polynomial). It is also clear that the functions $\hat{d}_{\text{sum}}$ and $\hat{d}_{\text{cov}}$ can be computed in polynomial time. Then, by Theorem 4, the reduction to SFM is complete.

▶ **Theorem 1.** *Sum-$k$-DMC and Cov-$k$-DMC can be solved in strongly polynomial time.*

Due to space limitations, we refer the reader to the full version [6] for details on designing $O(k^5n^5)$-time algorithms for these problems.

## 4 A Simple Algorithm for Finding Disjoint Minimum s-t Cuts

In the previous section, we looked at the problem of finding the $k$ most diverse minimum $s$-$t$ cuts in a graph. Here, we consider a slightly different problem. Observe that for diversity measures $d_{\text{sum}}$ and $d_{\text{cov}}$, the maximum diversity is achieved when the elements of a collection are all pairwise disjoint. Thus, it is natural to ask for a maximum cardinality collection of $s$-$t$ mincuts that are pairwise disjoint; i.e., that are as diverse as possible. We call this problem Maximum Disjoint Minimum $s$-$t$ Cuts (or Max-Disjoint MC for short).

**Max-Disjoint MC.** *Given a graph $G = (V, E)$ and vertices $s, t \in V(G)$, find a set $S \subseteq \Gamma_G(s,t)$ such that $X \cap Y = \emptyset$ for all $X, Y \in S$, and $|S|$ is as large as possible.*

Observe that a solution to Max-Disjoint MC immediately yields a solution to $k$-Disjoint Minimum $s$-$t$ Cuts. In this section, we prove Theorem 2 by giving an algorithm for Max-Disjoint MC that runs in $O(F(m,n) + \lambda(G)m)$ time, where $F(m,n)$ is the time required by a max-flow computation. First, we look at a restricted case when the input graph can be decomposed into a collection of edge-disjoint $s$-$t$ paths and (possibly) some additional edges – we refer to such a graph as an *$s$-$t$ path graph* – and devise an algorithm that handles such graphs. Then, we use this algorithm as a subroutine to obtain an algorithm that makes no assumption about the structure of the input graph.

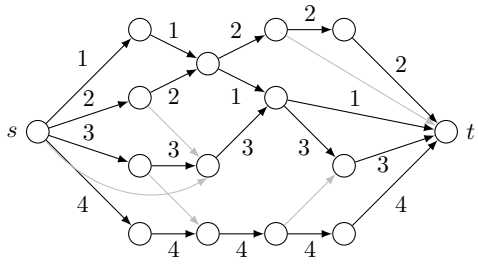


**Figure 1** Example of an $s$-$t$ path graph of height 4. Edges are labeled by integers corresponding to the path they belong to. Path edges are drawn in black and non-path edges in gray.

## 4.1 When the input is an s-t path graph

Let $H_{s,t}$ be a graph with designated vertices $s$ and $t$. We call $H_{s,t}$ an *$s$-$t$ path graph* (or *path graph* for short) if there is a collection $P$ of edge-disjoint $s$-$t$ paths such that $P$ covers all vertices in $V(H_{s,t})$. The *height* of $H_{s,t}$, denoted by $\lambda(H_{s,t})$, is the maximum number of edge-disjoint $s$-$t$ paths in the graph. For fixed $P$, we call the edges of $H_{s,t}$ in $P$ *path edges* and edges of $H_{s,t}$ not in $P$ *non-path edges*. Two vertices in $H_{s,t}$ are *path neighbors* if they are joined by a path edge, and *non-path neighbors* if they are joined (exclusively) by a non-path edge. See Figure 1 for an illustration.

Two remarks are in order. The first is that, by Menger's theorem, the size of a minimum $s$-$t$ cut in an $s$-$t$ path graph coincides with its height. The second remark is that, from a graph $G$, one can easily obtain a path graph $H_{s,t}$ of height $\lambda(G)$ by finding a maximum-sized set $\mathcal{P}_{s,t}$ of edge-disjoint $s$-$t$ paths in $G$ and letting $H_{s,t}$ be the induced subgraph of their union. Recall that, by Menger's theorem, a minimum $s$-$t$ cut in $G$ must contain exactly one edge from each path $p \in \mathcal{P}_{s,t}$. Thus, every minimum $s$-$t$ cut of $G$ is in $H_{s,t}$. However, the reverse is not always true. In the above construction, there could be multiple new minimum $s$-$t$ cuts introduced in $H_{s,t}$ that arise from ignoring the reachability between vertices of $\mathcal{P}_{s,t}$ in $G$. We will come back to this issue when discussing the general case in Section 4.2.

**The algorithm.** The goal in this subsection is to find a maximum cardinality collection $\hat{C}$ of pairwise disjoint $s$-$t$ mincuts in a path graph $H_{s,t}$. We now explain the main ideas behind the algorithm. Without loss of generality, assume that the underlying set of edge-disjoint $s$-$t$ paths that define $H_{s,t}$ is of maximum cardinality.

Let $X$ be an $s$-$t$ mincut in $H_{s,t}$, and suppose we are interested in finding an $s$-$t$ mincut $Y$ disjoint from $X$ such that $X < Y$. Consider any two edges $e = (u, u')$ and $f = (v, v')$ in $X$, and let $g = (w, w')$ be a path successor of $f$; that is $f \prec_p g$ with $p \in \mathcal{P}_{s,t}$. If there is a non-path edge $h = (u', z)$ such that $w' \leq z$, we say that $h$ is *crossing* w.r.t. $g$, and that $g$ is *invalid* w.r.t. $X$ (see Figure 2 for an illustration). The notions of crossing and invalid edges provide the means to identify the edges that cannot possibly be contained in $Y$. Let $E_{\text{inv}}(X)$ denote the set of invalid edges w.r.t. $X$. We make the following observation.

▶ **Observation 17.** *Let $Y > X$. Then $Y$ cannot contain an edge from $E_{\text{inv}}(X)$.*

**Proof.** For the sake of contradiction, suppose there exists an edge $g = (w, w')$ in $E_{\text{inv}}(X) \cap Y$. Consider the path $p_1 \in \mathcal{P}_{s,t}$, and let $f$ be the predecessor of $g$ on $p_1$ that is in $X$. Since $g \in E_{\text{inv}}(X)$, there is a crossing edge $h = (u', z)$ w.r.t. $g$. Let $p_2 \in \mathcal{P}_{s,t}$ be the path containing $u'$, and let $(u, u')$ be the edge of $p_2$ that is in $X$. Let $p_3$ be the s-t path that follows $p_2$ from $s$ to $u$, then follows the crossing edge $h$, and then continues along $p_1$ to $t$. Since $Y$ is an $s$-$t$

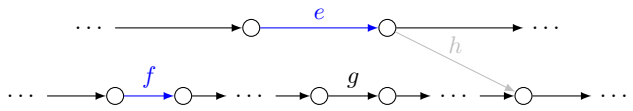


**Figure 2** Example illustrating the notions of crossing and invalid edges for an $s$-$t$ mincut $X$. Path and non-path edges are drawn in black and gray, respectively. Edges $e, f \in X$ are highlighted in blue. The edge $g$ is invalid w.r.t. $X$ since the edge $h$ is crossing with respect to it.

cut it must contain an edge from this path. Since $Y$ must contain exactly one edge from each path in $\mathcal{P}_{s,t}$, it cannot contain $h$. Moreover, $Y$ already contains edge $g$ from $p_1$. Then Y must contain an edge from the part of $p_2$ from $s$ to $u'$. But this contradicts our assumption that $Y > X$. ◀

If we extend the definition of $E_{\text{inv}}(X)$ to also include all the edges that are path predecessors of edges in $X$, we obtain that, for any $s$-$t$ path $p \in \mathcal{P}_{s,t}$, the set of invalid edges along $p$ is a prefix of the path. As a result, if we can identify the (extended) set $E_{\text{inv}}(X)$, then we can restrict our search of cut $Y$ to the set of valid edges $E_{\text{val}}(X) := E(H_{s,t}) \setminus E_{\text{inv}}(X)$. This motivates the following iterative algorithm for finding a pairwise disjoint collection of $s$-$t$ mincuts: (1) Find the leftmost $s$-$t$ mincut $X$ in $H_{s,t}$, (2) identify the set $E_{\text{inv}}(X)$ and find the leftmost $s$-$t$ mincut $Y$ amongst $E_{\text{val}}(X)$, (3) set $X = Y$ and repeat step (2) until $E_{\text{val}}(X) \cap p = \emptyset$ for any one path $p \in \mathcal{P}_{s,t}$, and finally (4) output the union of identified cuts as the returned collection $\hat{C}$. Informally, notice that the $s$-$t$ mincut identified at iteration $i$ is a successor of the mincuts identified at iterations $j < i$. Hence, the returned collection will consist of left-right ordered and pairwise disjoint $s$-$t$ mincuts. Moreover, picking the leftmost cut at each iteration prevents the set of invalid edges from growing unnecessarily large, which allows for more iterations and thus, a larger set returned. Next, we give a more formal description of the algorithm, the details of which are presented in Algorithm 1.

**Algorithm 1** Obtain a Maximum Set of Disjoint Minimum $s$-$t$ Cuts.

**Input:** Path graph $H_{s,t}$.
**Output:** A maximum set $\hat{C}$ of disjoint $s$-$t$ mincuts in $H_{s,t}$.

Initialize collection $\hat{C} \leftarrow \emptyset$ and set $M \leftarrow \{s\}$.
**while** $t$ is unmarked **do** ▷ Traverse the graph from left to right.
  **while** $M$ is not empty **do** ▷ Marking step.
    **for** each vertex $v \in M$ **do**
      **for** each path $p \in \mathcal{P}_{s,t}$ **do** ▷ Identify invalid edges.
        Identify the rightmost neighbor $u \in p$ of $v$ reachable by a non-path edge.
        **if** $u$ is unmarked **then**
          Mark $u$ and all (unmarked) vertices that are path-predecessors of $u$.
    Set $M$ to the set of newly marked vertices.
  $X \leftarrow \bigcup\{(x,y) \in \mathcal{P}_{s,t} : x \text{ is marked, } y \text{ is unmarked}\}$. ▷ Cut-finding step.
  $\hat{C} \leftarrow \hat{C} \cup \{X\}$.
  **for** each $(x,y) \in X$ **do** ▷ Mark the head node of cut edges.
    Mark $y$.
  $M \leftarrow \bigcup_{(x,y)\in X} y$. ▷ Newly marked vertices.
Return $\hat{C}$.

The algorithm works by traversing the graph from left to right in iterations while marking the vertices it visits. Initially, all vertices are unmarked, except for $s$. Each iteration consists of two parts: a *marking*, and a *cut-finding* step. In the marking step (Lines 3-9), the algorithm identifies currently invalid edges by marking the non-path neighbors – and their path-predecessors – of currently marked vertices. (Observe that a path edge becomes invalid if both of its endpoints are marked.) In Algorithm 1, this is realized by a variable $M$ that keeps track of the vertices that have just been marked as a consequence of the marking of vertices previously present in $M$. In the cut-finding step (Lines 10-14), the algorithm then finds the leftmost minimum $s$-$t$ cut amongst valid path edges. Notice that, for each $s$-$t$ path in $\mathcal{P}_{s,t}$, removing its first valid edge prevents $s$ from reaching $t$ via that path. This means that our leftmost cut of interest is the set of all path edges that have exactly one of their endpoints marked. Following the identification of this cut, the step concludes by marking the head vertices of the identified cut edges. Finally, the algorithm terminates when the target vertex $t$ is visited and marked. See Figure 3 for an example execution of the algorithm.

▷ Claim 18. The complexity of Algorithm 1 on an $m$-edge, $n$-vertex path graph is $O(m \log n)$.

Due to space limitations, we defer the proof of Claim 18 to the full version [6].

**Correctness of Algorithm 1.** We note an important property of collections of $s$-$t$ mincuts. (We use $d(C)$ to denote any of $d_{\text{sum}}(C)$ or $d_{\text{cov}}(C)$.)

▷ Claim 19. Let $C$ be a left-right ordered collection of minimum $s$-$t$ cuts in a graph $G$, the collection $\tilde{C}$ obtained by replacing $S_{\min}(\bigcup_{X \in C} X)$ (resp. $S_{\max}(\bigcup_{X \in C} X)$) with $S_{\min}(G)$ (resp. $S_{\max}(G)$) has cost $d(\tilde{C}) \leq d(C)$.

Proof. We prove this only for $S_{\min}(\cdot)$ as the proof for $S_{\max}(\cdot)$ is analogous. For simplicity, let us denote $S_{\min}(C) := S_{\min}(\bigcup_{X \in C} X)$. By definition, we know that no edge of $\bigcup_{X \in C} X$ lies to the left of $S_{min}(G)$. Then replacing $S_{\min}(C)$ with $S_{\min}(G)$ can only decrease the number of pairwise intersections previously present between $S_{\min}(C)$ and the cuts in $C \setminus S_{\min}(C)$. Notice that our measures of diversity only penalize edge intersections. Hence, the cost of collection $\tilde{C}$ cannot be greater than that of $C$. ◁

Now, consider an arbitrary collection of $k$ edge-disjoint $s$-$t$ mincuts in a path graph $H_{s,t}$. Corollary 6 implies that there also exists a collection of $k$ edge-disjoint $s$-$t$ mincuts in $H_{s,t}$ that is in left-right order. In particular, this is true for a collection of maximum cardinality $k_{\mathbf{max}}$. Together with Claim 19, this means that there always exists a collection $\hat{C}$ of edge-disjoint $s$-$t$ mincuts in $H_{s,t}$ with the following properties:

- **I** $\hat{C}$ has size $k_{\mathbf{max}}$,
- **II** $\hat{C}$ is in left-right order,
- **III** $\hat{C}$ contains the leftmost $s$-$t$ mincut of $H_{s,t}$, and
- **IV** The set $S_{\max}(\hat{C}) \cap S_{\max}(H_{s,t})$ is not empty.

We devote the rest of the subsection to proving the following lemma, which serves to prove the correctness of Algorithm 1.

▶ **Lemma 20.** *Algorithm 1 returns a collection of edge-disjoint minimum $s$-$t$ cuts that satisfies Properties I–IV.*

Let $\hat{C}$ denote the solution returned by the algorithm. First, we show that $\hat{C}$ contains only disjoint cuts. This follows from the fact that a cut can only be found amongst valid edges at any given iteration, and once an edge has been included in a cut, it becomes invalid

at every subsequent iteration. Similarly, Properties II and III are consequences of the notion of invalid edges. We start by proving the latter. Let $X_1$ denote the leftmost cut in $\hat{C}$. For the sake of contradiction, assume there is a minimum $s$-$t$ cut $Y$ such that $e \prec_p f$. Here, $e \in Y$, $f \in X_1$ and *w.l.o.g.* $p$ is an $s$-$t$ path from any arbitrary maximum collection of $s$-$t$ paths in $H_{s,t}$. For the algorithm to pick edge $f = (u, u')$ as part of $X_1$ it must be that vertex $u$ is marked and $u'$ is not. We know that the predecessors of marked vertices must also be marked. Hence we know that both endpoints of edge $e$ are marked. But by definition, this means that edge $e$ is invalid, and cannot be in a minimum $s$-$t$ cut. This gives us the necessary contradiction, and $X_1$ must be the leftmost cut in the graph.

We continue with Property II. This property follows from the fact that, at any given iteration, the posets of invalid path-edges on each path of $H_{s,t}$ are ideals of the set of path edges. This means that the edges in the cut found by the algorithm at iteration $i$ are all path predecessors of an edge in the cut found at iteration $i+1$. Carrying on with Property IV, we prove that it follows from the fact that the algorithm terminates when the target node $t$ is marked. Suppose, for the sake of contradiction, that the cuts $S_{\max}(\hat{C})$ and $S_{\max}(H_{s,t})$ do not intersect. Then, given that $S_{\max}(\hat{C})$ is the last cut found by our algorithm, to mark node $t$ there must exist a non-path edge connecting the endpoint $v$ of some edge $e = (u, v) \in S_{\max}(\hat{C})$ to $t$. But this implies that no path-successor of edge $e$ can be in an $s$-$t$ mincut, which makes $e$ the rightmost edge on its path that belongs to an $s$-$t$ mincut. Therefore, $e$ must also be contained in $S_{\max}(H_{s,t})$, a contradiction.

It only remains to show Property I, which states that the collection $\hat{C}$ is of maximum cardinality $k_{\mathbf{max}}$. For this, we make the following claim, whose proof is analogous to the proof of Property III. Let $\hat{C}_i$ be the collection of $s$-$t$ mincuts maintained by the algorithm at the end of iteration $i$.

▷ Claim 21. Consider set $\hat{C}_{i-1}$ and let $X_i$ be the minimum $s$-$t$ cut found by the algorithm at iteration $i$. Then, there is no minimum $s$-$t$ cut $Y$ such that: (i) $Y$ is disjoint from each $X \in \hat{C}_{i-1}$, and (ii) $Y$ contains an edge that is a path predecessor of an edge of $X_i$.

In other words, as the algorithm makes progress, no minimum $s$-$t$ cut – that is disjoint from the ones found so far by the algorithm – has edges to the left of the minimum $s$-$t$ cut found by the algorithm at the present iteration. Next, we show that this implies the maximality of the size of the solution returned by the algorithm.

Let $C_{\max}$ be a maximum-sized collection of $s$-$t$ mincuts in the graph. Without loss of generality, assume that $C_{\max}$ is in left-right order (otherwise, by Corollary 6 we can always obtain an equivalent collection that is left-right ordered) and that $S_{\min}(H_{s,t}) \in C_{\max}$ and $S_{\max}(H_{s,t}) \in C_{\max}$. For the sake of contradiction, suppose that the collection $\hat{C}$ returned by our algorithm is of cardinality $|\hat{C}| = \ell < k_{\max}$.

▶ **Observation 22.** *There exists at least one minimum $s$-$t$ cut $Y \in C_{\max}$ such that $X_i < Y$ and $Y$ contains at least one edge that is a path predecessor of an edge in $X_{i+1}$, with $X_i$ and $X_{i+1}$ two consecutive cuts in $\hat{C}$.*

**Proof.** Let $C_{\max} = \{Y_1, Y_2, \ldots, Y_{k_{\max}}\}$, where $Y_1 = S_{\min}(H_{s,t})$ and $Y_{k_{\max}} = S_{\max}(H_{s,t})$, and let $\hat{C} = \{X_1, \ldots, X_\ell\}$. We know by Property III that $X_1 = S_{\min}(H_{s,t})$. Hence, there is always an $s$-$t$ mincut in $C_{\max}$ that is a strict successor of a cut in $\hat{C}$, namely $Y_2 > X_1$. For the sake of contradiction, suppose that the observation is false. Then, every cut $Y \in C_{\max}$ that is a strict successor of a cut $X_i \in \hat{C}$ is also a (not necessarily strict) successor of the cut $X_{i+1} \in \hat{C}$, for $i \in \{1, \ldots, \ell - 1\}$. Let this be true for the first $\ell - 1$ cuts of $\hat{C}$. Then, the last $k_{\max} - \ell$ cuts of $C_{\max}$ must be disjoint from the first $\ell - 1$ cuts of $\hat{C}$. The last cut $X_\ell$ of $\hat{C}$

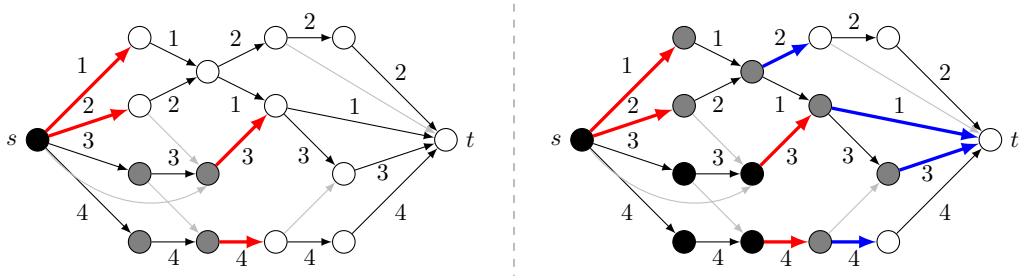


**Figure 3** Example illustrating the first two iterations of Algorithm 1 on a path graph of height 4. The black- and gray-shaded vertices represent vertices marked at the previous and current iterations, respectively. The red edges correspond to the $s$-$t$ mincut found at the end of the first (left) iteration. Similarly, the blue edges correspond to the $s$-$t$ mincut found at the second (right) iteration.

must then be located in or before the gap between the first $\ell$ cuts in $C_{\max}$ and its remaining $k - \ell$ cuts. But we know by Property IV that $X_\ell \cap Y_{k_{\max}} \neq \emptyset$, which gives the necessary contradiction. ◀

Observation 22 stands in contrast with Claim 21, which states that such a cut $Y$ cannot exist. Hence, we obtain a contradiction, and the collection $\hat{C}$ returned by the algorithm must be of maximum cardinality. This completes the proof of Lemma 20.

## 4.2 Handling the general case

We now consider Max-Disjoint MC in general graphs. Recall from the previous subsection that, from a graph $G$, one can construct a path graph $H_{s,t}$ such that every minimum $s$-$t$ cut in $G$ is also a minimum $s$-$t$ cut in $H_{s,t}$. We could propose to use Algorithm 1 in $H_{s,t}$ to solve Max-Disjoint MC in $G$. But, as we argued previously, the path graph $H_{s,t}$ may not have the same set of $s$-$t$ mincuts as $G$. We can, however, solve this challenge by augmenting $H_{s,t}$ with edges such that its minimum $s$-$t$ cuts correspond bijectively to those in $G$.

▶ **Definition 23.** An *augmented $s$-$t$ path graph* of $G$ is a path graph $H_{s,t}(G)$ of height $\lambda(G)$, with additional non-path edges between any two vertices $u, v \in V(H_{s,t}(G))$ such that $v$ is reachable from $u$ in $G$ by a path whose internal vertices are exclusively in $V(G) \setminus V(H_{s,t}(G))$.

In view of this definition, the following claim and lemma serve as the correctness and complexity proofs of the proposed algorithm for the general case.

▷ Claim 24. An augmented $s$-$t$ path graph of $G$ has the same set of $s$-$t$ mincuts as $G$.

Proof. By Menger's theorem, we know that a minimum $s$-$t$ cut in $G$ must contain exactly one edge from each path in $\mathcal{P}_{s,t}(G)$, where $|\mathcal{P}_{s,t}(G)| = |\lambda(G)|$. W.l.o.g., let $H_{s,t}(G)$ be the augmented $s$-$t$ path graph of $G$ such that each path $p \in \mathcal{P}_{s,t}(G)$ is also present in $H_{s,t}(G)$. We now show that a minimum $s$-$t$ cut in $G$ is also present in $H_{s,t}(G)$. The argument in the other direction is similar and is thus omitted.

Consider an arbitrary minimum $s$-$t$ cut $X$ in $G$. For the sake of contradiction, assume that $X$ is not a minimum $s$-$t$ cut in $H_{s,t}(G)$. Then, after removing every edge of $X$ in $H_{s,t}(G)$, there is still at least one $s$-$t$ path left in the graph. Such a path must contain an edge $(u, v)$ such that $u \leq w$ and $w' \leq v$, where $w$ and $w'$ are the tail and head nodes of two (not necessarily distinct) edges in $X$, respectively. By definition of $H_{s,t}(G)$, there is a path from $u$ to $v$ in $G$ that does not use edges in $\mathcal{P}_{s,t}(G)$. But then removing the edges of $X$ in $G$ still leaves an $s$-$t$ path in the graph. Thus $X$ cannot be an $s$-$t$ cut, and we reach our contradiction. ◁

▶ **Lemma 25.** *An augmented $s$-$t$ path graph $H$ of a graph $G$ can be constructed in time $O(F(m,n) + m\lambda(G))$, where $F(m,n)$ is the time required by a max-flow computation.*

**Proof.** The idea of the algorithm is rather simple. First, we find a maximum cardinality collection of edge-disjoint $s$-$t$ paths in $G$ and take their union to construct a "skeleton" graph $H$. Then, we augment the graph by drawing an edge between two path vertices $u, v \in H$ if $v$ is reachable from $u$ in $G$ by using exclusively non-path vertices. By definition, the resulting graph is an augmented $s$-$t$ path graph of $G$.

Now we look into the algorithm's implementation and analyze its running time. It is folklore knowledge that the problem of finding a maximum-sized collection of edge-disjoint $s$-$t$ paths in a graph with $n$ vertices and $m$ edges can be formulated as a maximum flow problem. Hence, the first step of the algorithm can be performed in $F(m,n)$ time. Let $\mathcal{P}_{s,t}(G)$ denote such found collection of $s$-$t$ paths.

The second step of the algorithm could be computed in $O(mn)$ time by means of an *all-pairs reachability* algorithm. Notice, however, that for a path vertex $v$ all we require for a correct execution of Algorithm 1 is knowledge of the rightmost vertices it can reach on each of the $\lambda(G)$ paths (Line 6 of Algorithm 1). Hence, we do not need to draw every edge between every pair of reachable path vertices, only $\lambda(G)$ edges per vertex suffice. This can be achieved in $O(m\lambda(G))$ time as follows.

In the original graph, first equip each vertex $u \in V(G)$ with a set of $\lambda(G)$ variables $R(p,u)$, one for each path $p \in \mathcal{P}_{s,t}(G)$. These variables will be used to store the rightmost vertex $v \in p$ that is reachable from $u$. Next, consider a path $p \in \mathcal{P}_{s,t}(G)$ represented as a sequence $[v_1, v_2, \ldots, v_p]$ of internal vertices (i.e., with $s$ and $t$ removed). For each vertex $v \in p$, in descending order, execute the following procedure `propagate(`$v$`, `$p$`)`: Find the set $N(v)$ of incoming neighbors of $v$ in $G$ and, for each $w \in N(v)$ if $R(p,w)$ has not been set, let $R(p,w) = v$ and mark $w$ as visited. Then, for each node $w \in N(v)$, if $w$ is an unvisited non-path vertex, execute `propagate(`$w$`, `$p$`)`; otherwise, do nothing. Notice that, since we iterate from the rightmost vertex in $p$, any node $u$ such that $R(u,p) = v_i$ cannot change its value when executing `propagate(`$v_j$`)` with $j < i$. In other words, each vertex only stores information about the rightmost vertex it can reach in $p$. Complexity-wise, every vertex $v$ in $G$ will be operated upon at most $deg(v)$ times. Hence, starting from an unmarked graph, a call to `propagate(`$v$`, `$p$`)` takes $O(m)$ time. Now, we want to execute the above for each path $p \in \mathcal{P}_{s,t}(G)$ (unmarking all vertices before the start of each iteration). This then gives us our claimed complexity of $O(m\lambda(G))$. The claim follows from combining the running time of both steps of the algorithm. ◀

The following is an immediate consequence of Lemma 25 and Claim 18.

▶ **Corollary 26.** *There is an algorithm that, given a graph $G$ and two specified vertices $s, t \in V(G)$, in $O(F(m,n) + m\lambda(G))$ time finds a collection of maximum cardinality of pairwise disjoint $s$-$t$ mincuts in $G$.*

By replacing $F(m,n)$ in Corollary 26 with the running time of the current best algorithms for finding a maximum flow [21, 18], we obtain the desired running time of Theorem 2.

## 5 Concluding remarks

We showed that $k$-DMC can be solved efficiently when considering two natural measures for the diversity of a set of solutions. There exist, however, other sensible measures of diversity. One that often arises in literature is that of maximizing the minimum pairwise Hamming distance of a solution set. The complexity of $k$-DMC when considering this objective is still open. It is not immediately clear how to apply our ordering results to this variant.

For the special case of $k$-Disjoint Minimum $s$-$t$ Cuts, we showed that faster algorithms exist when compared to solving $k$-DMC on the total-sum and coverage diversity measures. It is thus natural to ask whether there are faster algorithms for Sum-$k$-DMC and Cov-$k$-DMC (or other variants of $k$-DMC) that do not require the sophisticated framework of submodular function minimization. In this work, we relied on the algebraic structure of the problem to obtain a polynomial time algorithm. We believe it is an interesting research direction to assess whether the notion of diversity in other combinatorial problems leads to similar structures, which could then be exploited for developing efficient algorithms.

## References


1 Julien Baste, Michael R. Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303:103644, 2022. doi:10.1016/j.artint.2021.103644.

2 Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. Fpt algorithms for diverse collections of hitting sets. *Algorithms*, 12(12):254, 2019.

3 Garrett Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, 1937.

4 Mohammadreza Bolandnazar, Woonghee Tim Huh, S Thomas McCormick, and Kazuo Murota. A note on "order-based cost optimization in assemble-to-order systems". *University of Tokyo (February, Techical report)*, 2015.

5 Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge University Press, 2002.

6 Mark de Berg, Andrés López Martínez, and Frits Spieksma. Finding diverse minimum s-t cuts, 2023. arXiv:2303.07290.

7 Fernando Escalante. Schnittverbände in graphen. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 38, pages 199–220. Springer, 1972.

8 Fedor V. Fomin, Petr A. Golovach, Lars Jaffke, Geevarghese Philip, and Danil Sagunov. Diverse Pairs of Matchings. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2020.26.

9 George Gratzer. *Lattice theory: First concepts and distributive lattices*. Courier Corporation, 2009.

10 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer Science & Business Media, 2012.

11 D. Gusfield and R.W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. Foundations of computing. MIT Press, 1989.

12 R Halin. Lattices related to separation in graphs. In *Finite and Infinite Combinatorics in Sets and Logic*, pages 153–167. Springer, 1993.

13 Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3968–3976, 2023.

14 Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, See Woo Lee, and Yota Otachi. Computing diverse shortest paths efficiently: A theoretical and experimental study. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):3758–3766, June 2022. doi:10.1609/aaai.v36i4.20290.

15 Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3778–3786, 2021.

**16** Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

**17** Haotian Jiang. Minimizing convex functions with integral minimizers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 976–985. SIAM, 2021.

**18** Tarun Kathuria. A potential reduction inspired algorithm for exact max flow in almost $\widetilde{O}(m^{4/3})$ time, 2020. `arXiv:2009.03260`.

**19** Samir Khuller, Joseph Naor, and Philip Klein. The lattice structure of flow in planar graphs. *SIAM Journal on Discrete Mathematics*, 6(3):477–490, 1993.

**20** Ching-Chung Kuo, Fred Glover, and Krishna S Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185, 1993.

**21** Yang P. Liu and Aaron Sidford. Faster divergence maximization for faster maximum flow, 2020. `arXiv:2003.08929`.

**22** George Markowsky. An overview of the poset of irreducibles. *Combinatorial And Computational Mathematics*, pages 162–177, 2001.

**23** Bernd Meyer. On the lattices of cutsets in finite graphs. *European Journal of Combinatorics*, 3(2):153–157, 1982.

**24** Kazuo Murota. *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics, 2003. `doi:10.1137/1.9780898718508`.

**25** Jean-Claude Picard and Maurice Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Studies*, 13:8–16, 1980.

**26** Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

**27** Shengling Shi, Xiaodong Cheng, and Paul M.J. Van den Hof. Generic identifiability of subnetworks in a linear dynamic network: The full measurement case. *Automatica*, 137:110093, 2022. `doi:10.1016/j.automatica.2021.110093`.

**28** Shengling Shi, Xiaodong Cheng, and Paul M.J. Van den Hof. Personal communication, October 2021.

**29** Donald K Wagner. Disjoint (s, t)-cuts in a network. *Networks*, 20(4):361–371, 1990.

**30** Si-Qing Zheng, Bing Yang, Mei Yang, and Jianping Wang. Finding minimum-cost paths with minimum sharability. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 1532–1540. IEEE, 2007.