A Framework to Design Approximation Algorithms for Finding Diverse Solutions in Combinatorial Problems

Tesshu Hanaka[1], **Masashi Kiyomi**[2], **Yasuaki Kobayashi**[3], **Yusuke Kobayashi**[4], **Kazuhiro Kurita**[5], **Yota Otachi**[5]

[1] Kyushu University, Fukuoka, Japan
[2] Seikei University, Musashino-shi, Tokyo, Japan
[3] Hokkaido University, Sapporo, Japan
[4] Kyoto University, Kyoto, Japan
[5] Natoya University, Nagoya, Japan
hanaka@inf.kyushu-u.ac.jp, kiyomi@st.seikei.ac.jp, koba@ist.hokudai.ac.jp, yusuke@kurims.kyoto-u.ac.jp, kurita@i.nagoya-u.ac.jp, otachi@nagoya-u.jp

Abstract

Finding a *single* best solution is the most common objective in combinatorial optimization problems. However, such a single solution may not be applicable to real-world problems as objective functions and constraints are only "approximately" formulated for original real-world problems. To solve this issue, finding *multiple* solutions is a natural direction, and diversity of solutions is an important concept in this context. Unfortunately, finding diverse solutions is much harder than finding a single solution. To cope with the difficulty, we investigate the approximability of finding diverse solutions. As a main result, we propose a framework to design approximation algorithms for finding diverse solutions, which yields several outcomes including constant-factor approximation algorithms for finding diverse matchings in graphs and diverse common bases in two matroids and PTASes for finding diverse minimum cuts and interval schedulings.

Introduction

One way to solve a real-world problem is to formulate the problem as a mathematical optimization problem and find a solution with an optimization algorithm. However, it is not always easy to formulate an appropriate optimization problem as real-world problems often include intricate constraints and implicit preferences, which are usually simplified in order to solve optimization problems. Hence, an optimal solution obtained in this way is not guaranteed to be a "good solution" to the original real-world problem. To cope with this underlying inconsistency, the following two-stage approach would be promising: algorithms find multiple solutions and then users find what they like from these solutions. One may think that top-k enumeration algorithms (see [Eppstein 2008] for a survey) can be used for this purpose. However, this is not always the case since top-k enumeration algorithms may output solutions similar to one another. (See [Wang, Cheng, and Fu 2013; Yuan et al. 2016; Hao, Pei, and Yang 2020], for example). Such a set of solutions are not useful as a "catalog" of solutions provided to users.

As a way to resolve this issue, algorithms are expected to find "diverse" solutions, and then finding "diverse" solutions has received considerable attention in several fields such as Artificial Intelligence [Ingmar et al. 2020; Nadel 2011], Data Mining [Wang, Cheng, and Fu 2013; Yuan et al. 2016], and Operations Research [Danna and Woodruff 2009; Petit and Trapp 2019]. The problem of finding diverse solutions can be modeled as a multi-objective optimization problem, which optimizes some diversity measure and the quality of solutions simultaneously. To solve this multi-objective optimization problem, there are several approaches, such as mathematical programming [Danna et al. 2007; Danna and Woodruff 2009; Petit and Trapp 2019], constraint programming [Hebrard et al. 2005; Petit and Trapp 2015], heuristics [Danna et al. 2007; Drosou and Pitoura 2010; Hentenryck, Coffrin, and Gutkovich 2009; Vieira et al. 2011], and so forth. See Table 1 in [Petit and Trapp 2019], which summarizes various approaches to find diverse solutions in the literature. These approaches are not only practical but also versatile, enabling us to formulate various combinatorial problems in these approaches. However, theoretical guarantees on the running time of algorithms and/or the quality of solutions would be difficult to obtain as some (general purpose) mathematical/constraint solvers or heuristic objectives/algorithms are key components of these algorithms.

Recently, theoretical aspects of the problem of finding diverse solutions in combinatorial problems are investigated. This research direction would be made by Fellows and Rosamond who proposed the *diverse X paradigm* in Dagstuhl Seminar 18421 [Fernau et al. 2019]. In this paradigm, "X" is a placeholder that represents solutions we are looking for, and they asked for theoretical investigations of finding diverse solutions. Since the problem of finding diverse solutions is much harder than that of finding a single solution for some "X", it would be reasonable to consider the problem from the perspective of fixed-parameter tractability[1]. From this proposition, several fixed-parameter tractable (FPT) algorithms are developed so far. Baste et al. gave

[1]Roughly speaking, the goal is to develop algorithms that run in time $f(k)\mathrm{poly}(n)$, where n is the input size and k is a parameter defined on a specific problem.

algorithms for finding diverse solutions related to hitting sets [Baste et al. 2019] and those on bounded-treewidth graphs [Baste et al. 2022]. Hanaka et al. [Hanaka et al. 2021] proposed a framework to obtain FPT algorithms for finding diverse solutions in various combinatorial problems. Fomin et al. [Fomin et al. 2020, 2021] investigated the fixed-parameter tractability of finding diverse solutions related to matchings and matroids. In these work, the number of solutions to be found is considered as a small parameter, which can be a potential drawback in practice. As we discussed, a set of diverse solutions would be displayed as a "catalog" of solutions and hence moderate number of solutions are essential to users to make their own decisions based on the displayed solutions.

For this reason, we aim to develop theoretically efficient algorithms for finding a moderate number of diverse solutions rather than a small number of diverse solutions. As we mentioned, the problem of finding diverse solutions is harder than that of finding a single solution. We first observe that diversity measures have a significant impact on the computational complexity of the diverse version of combinatorial problems: The problem of computing k bases of a matroid maximizing the minimum (weighted) Hamming distance (MAX-MIN HAMMING DISTANCE) is NP-hard [Fomin et al. 2021], while the problem maximizing the sum of (weighted) Hamming distance (MAX-SUM HAMMING DISTANCE) is solvable in polynomial time [Hanaka et al. 2021]. Hanaka et al. [Hanaka et al. 2022] enhanced this observation by showing that the diverse versions of several classical combinatorial problems, such as bipartite matchings, arborescences, shortest paths, are polynomial-time solvable under MAX-SUM HAMMING DISTANCE, while no such results for MAX-MIN HAMMING DISTANCE are known. These circumstances indicate that MAX-SUM HAMMING DISTANCE is theoretically easier than MAX-MIN HAMMING DISTANCE. However, there are still computationally hard problems under MAX-SUM HAMMING DISTANCE: For example, the problem of computing a maximum matching in a graph is known to be solvable in polynomial time, whereas that of computing two maximum matchings M_1 and M_2 maximizing $|M_1 \triangle M_2|$ is known to be NP-hard [Holyer 1981] (see Section 7 for other examples). Thus, we tackle this intractability by developing *polynomial-time approximation algorithms* for the diverse version of various combinatorial problems. To this end, we employ MAX-SUM HAMMING DISTANCE as our diversity measure (see **??** for its definition), which might be somewhat theoretically tractable, but there are still several obstacles to be overcome. We note that this diversity measure is frequently used in both experimental and theoretical settings [Baste et al. 2022; Danna and Woodruff 2009; Hanaka et al. 2022; Hentenryck, Coffrin, and Gutkovich 2009; Petit and Trapp 2015]

Our main result is a framework for designing efficient approximation algorithms with constant approximation factors for finding diverse solutions in combinatorial problems. Roughly speaking, our approximation framework says that if we can *enumerate* top-k *weighted* solutions in polynomial time, then we can obtain in polynomial time *unweighted* solutions maximizing our diversity measure with constant approximation factors. Moreover, suppose that we can exactly maximize our diversity of solutions in polynomial time when the number of solutions we are looking for is bounded by a constant. Then, our framework yields a polynomial-time approximation scheme (PTAS), meaning that factor-$(1 - \varepsilon)$ approximation in polynomial time for every constant $\varepsilon > 0$. By applying our framework, we obtain efficient constant-factor approximation algorithms for finding diverse matchings in a graph and common bases of two matroids, while PTASes for finding diverse minimum cuts and interval schedulings. Let us note that these diversity maximization problems are unlikely to be solvable in polynomial time, which will be discussed later.

The approximation factor of our framework comes from previous work on the dispersion problem (see **??** for its definition). A similar framework was independently proposed by Gao et al. [Gao et al. 2022]. In both frameworks, the subproblem of finding a "furthest solution" is a key ingredient. They reduced this subproblem to the budget-constrained optimization problem and solve it by bi-approximation algorithms. This makes their framework more flexible, allowing to find diverse approximate weighted solutions, while our framework only focus on unweighted solutions. As opposed to this weight restriction, the approximation factor of our framework is much better than theirs: their approximation factor is $1/2$, while ours are $\max(1-1/k, 1/2)$ or even $1-\varepsilon$ for any constant $\varepsilon > 0$, where k is the number of solutions we are looking for.

Due to the space limitation, some proofs (marked $\star$) are deferred to Appendix.

The rest of this paper is organized as follows. The next section gives some notation and terminology used in this paper. In particular, we give an overview of the result of [Cevallos, Eisenbrand, and Zenklusen 2019], which is a key to our approximation algorithms. In **??** , we describe our framework to find diverse solutions in combinatorial problems. Then, in **??** , we discuss some applications of our framework, including the diverse versions of the maximum matching problem, the matroid intersection problem, the minimum cut problem, and the interval scheduling problem. Finally, we conclude our paper with some further directions in **??** .

Preliminaries

We denote the set of real numbers, the set of non-negative real numbers, and the set of positive real numbers as $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$, respectively. Let E be a set. We denote the set of all subsets of E as 2^E. A function $d\colon E \times E \to \mathbb{R}_{\geq 0}$ is called a *metric* (on E) if it satisfies the following conditions: for $x, y, z \in E$, (1) $d(x, y) = 0$ if and only if $x = y$; (2) $d(x, y) = d(y, x)$; (3) $d(x, z) \leq d(x, y) + d(y, z)$. Suppose that $E \subseteq \mathbb{R}^m$ for some integer m. For $x \in E$, we denote by x_i the ith component of x. If $d(x, y) = \sum_{1 \leq i \leq m} |x_i - y_i|$ holds for $x, y \in E$, then d is called an ℓ_1-*metric*.

Let E be a finite set. For $X, Y \subseteq E$, the symmetric difference between X and Y is denoted by $X \triangle Y$ (i.e., $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$). Let $w : E \to \mathbb{R}_{\geq 0}$. A *weighted Hamming distance* is a function $d : 2^E \times 2^E \to \mathbb{R}_{\geq 0}$ such that for $X, Y \subseteq E$, $d_w(X,Y) = w(X \triangle Y)$, where $w(Z) = \sum_{x \in Z} w(x)$ for $Z \subseteq E$. Suppose that $E = \{1, 2, \ldots, m\}$. We can regard each subset $X \subseteq E$ as an m-dimensional vector $x = (x_1, \ldots, x_m)$ defined by $x_i = w(i)$ if $i \in X$ and $x_i = 0$ otherwise, for $1 \leq i \leq m$. It is easy to observe that for $X, Y \subseteq E$, $d_w(X,Y) = \sum_{1 \leq i \leq m} |x_i - y_i|$, where x and y are the vectors corresponding to X and Y, respectively. Thus, the weighted Hamming distance d_w can be considered as an ℓ_1-metric.

In this paper, we focus on the following diversity measure $d_{\mathrm{sum}}(\cdot)$, called the *sum diversity*. Let $\mathcal{Y} = \{Y_1, \ldots, Y_k\}$ be a collection of subsets of E and $w\colon E \to \mathbb{R}_{\geq 0}$ be a weight function. We define $d_{\mathrm{sum}}(\mathcal{Y}) = \sum_{1 \leq i < j \leq k} d_w(Y_i, Y_j)$.

Our problem MAX-SUM DIVERSE SOLUTIONS is defined as follows.

Definition 1 (MAX-SUM DIVERSE SOLUTIONS)**.** *Given a finite set E, an integer k, a weight function $w\colon E \to \mathbb{R}_{\geq 0}$, and a membership oracle for $\mathcal{X} \subseteq 2^E$, the task of* MAX-SUM DIVERSE SOLUTIONS *is to find a set $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_k\}$ of k distinct subsets $Y_1, Y_2, \ldots, Y_k \in \mathcal{X}$ that maximizes the sum diversity $d_{\mathrm{sum}}(\mathcal{Y})$.*

Each set in $\mathcal{X}$ is called a *feasible solution*. In MAX-SUM DIVERSE SOLUTIONS, the set $\mathcal{X}$ of feasible solutions is not given explicitly, while we can test whether a set $X \subseteq E$ belongs to $\mathcal{X}$. Our problem MAX-SUM DIVERSE SOLUTIONS is highly related to the problem of packing disjoint feasible solutions.

Observation 2. *Suppose that all sets in $\mathcal{X}$ have the same cardinality r and $w(e) = 1$ for $e \in E$. Let $Y_1, Y_2, \ldots, Y_k \in \mathcal{X}$ be k distinct subsets. Then, $d_{\mathrm{sum}}(\{Y_1, \ldots, Y_k\}) \geq kr(k-1)$ if and only if $Y_i \cap Y_j = \emptyset$ for $1 \leq i < j \leq k$.*

This observation implies several hardness results of MAX-SUM DIVERSE SOLUTIONS, which will be discussed in **??** .

We particularly focus on the approximability of MAX-SUM DIVERSE SOLUTIONS for specific sets of feasible solutions. For a maximization problem, we say that an approximation algorithm has factor $0 < \alpha \leq 1$ if given an instance I, the algorithm outputs a solution with objective value $\mathrm{ALG}(I)$ such that $\mathrm{ALG}(I)/\mathrm{OPT}(I) \geq \alpha$, where $\mathrm{OPT}(I)$ is the optimal value for I. A *polynomial-time approximation scheme* is an approximation algorithm that takes an instance I and a constant $\varepsilon > 0$, the algorithm outputs a solution with $\mathrm{ALG(I)/OPT(I)} \geq 1 - \varepsilon$ in polynomial time.

A Technique for MAX-SUM DIVERSIFICATION

Our framework is based on approximation algorithms for a similar problem MAX-SUM DIVERSIFICATION. Let X be a set and let $d\colon X \times X \to \mathbb{R}_{\geq 0}$ be a metric. In what follows, for $Y \subseteq X$, we denote $\sum_{x,y \in Y} d(x,y)$ as $d(Y)$.

Definition 3 (MAX-SUM DIVERSIFICATION)**.** *Given a metric $d\colon X \times X \to \mathbb{R}_{\geq 0}$ on a finite set X and an integer k, the task of* MAX-SUM DIVERSIFICATION *is to find a subset $Y \subseteq X$ with $|Y| = k$ that maximizes $d(Y)$.*

MAX-SUM DIVERSIFICATION is studied under various names such as MAX-AVG FACILITY DISPERSION and REMOTE-CLIQUE [Cevallos, Eisenbrand, and Zenklusen 2019; Ravi, Rosenkrantz, and Tayi 1994]. MAX-SUM DIVERSIFICATION is known to be NP-hard [Ravi, Rosenkrantz, and Tayi 1994]. Cevallos et al. [Cevallos, Eisenbrand, and Zenklusen 2019] devised a PTAS for MAX-SUM DIVERSIFICATION. Their algorithm is based on a rather simple local search technique, but their analysis of the approximation factor and the iteration bound are highly non-trivial. Our framework is based on their algorithm, which is briefly sketched below.

Algorithm 1: A $(1 - 2/k)$-approximation algorithm for MAX-SUM DIVERSIFICATION.

```
1 Procedure LocalSearch($X, d, k$)
2   $Y \leftarrow$ arbitrary $k$ elements in $X$;
3   for $i = 1, \ldots, \lceil \frac{k(k-1)}{(k+1)} \ln(\frac{(k+2)(k-1)^2}{4}) \rceil$ do
4     if $\exists$ pair $(x,y) \in (X \setminus Y) \times Y$ such that $d(Y - y + x) > d(Y)$ then
5       $(x,y) \leftarrow \operatorname{arg\,max}_{(x,y) \in (X \setminus Y) \times Y} d(Y - y + x)$;
6       $Y \leftarrow Y - y + x$;
7   Output $Y$;
```

A pseudocode of the algorithm due to [Cevallos, Eisenbrand, and Zenklusen 2019] is given in Algorithm 1. In this algorithm, we first pick an arbitrary set of k elements in X, which is denoted by $Y \subseteq X$. Then, we find a pair of elements $x \in X \setminus Y$ and $y \in Y$ that maximizes $d(Y - y + x)$ and update Y by $Y - y + x$ if $d(Y - y + x) > d(Y)$. We repeat this update procedure $\lceil \frac{k(k-1)}{(k+1)} \ln(\frac{(k+2)(k-1)^2}{4}) \rceil = O(k \log k)$ times. Since we can find a pair (x,y) in $O(|X|\, k\tau)$ time, where τ is the running time to evaluate the distance function $d(x,y)$ for $x, y \in X$, the following lemma holds.

Lemma 4. *Algorithm 1 runs in time $O(|X|\, k^2 \tau \log k)$.*

They showed that if the metric d is *negative type*, then the approximation ratio of Algorithm 1 is at least $1 - 2/k$ [Cevallos, Eisenbrand, and Zenklusen 2019]. Here, we do not give the precise definition of a negative type metric but mention that every ℓ_1-metric is negative type [Deza and Laurent 1997; Cevallos, Eisenbrand, and Zenklusen 2016].

Theorem 5 ([Cevallos, Eisenbrand, and Zenklusen 2019])**.** *If $d\colon X \times X \to \mathbb{R}_{\geq 0}$ is a negative type metric, then the approximation ratio of Algorithm 1 is $1 - 2/k$.*

They further observed that the above theorem implies that MAX-SUM DIVERSIFICATION admits a PTAS as follows. Let ϵ be a positive constant. When $\epsilon < 2/k$, that is, $k < 2/\epsilon$, then k is constant. Thus, we can solve MAX-SUM DIVERSIFICATION in time $|X|^{O(1/\epsilon)}$ by using a brute-force search. Otherwise, the above $(1 - 2/k)$-approximation algorithm achieves factor $1 - \epsilon$. Thus, MAX-SUM DIVERSIFICATION admits a PTAS, provided that d is a negative type metric.

Corollary 6 ([Cevallos, Eisenbrand, and Zenklusen 2019])**.** *If $d\colon X \times X \to \mathbb{R}_{\geq 0}$ is a negative type metric, then* MAX-SUM DIVERSIFICATION *admits a PTAS.*

A Framework for Finding Diverse Solutions

In this section, we propose a framework for designing approximation algorithms for MAX-SUM DIVERSE SOLUTIONS. The basic strategy to our framework is the local search algorithm described in the previous section. Let E be a finite set and let $\mathcal{X} \subseteq 2^E$ be a set of feasible solutions. We set $X := \mathcal{X}$ and apply the local search algorithm for MAX-SUM DIVERSIFICATION to (X, d_w, k). Recall that our diversity measure d_{sum} is the sum of weighted Hamming distances d_w. Moreover, d_w is an ℓ_1-metric, as observed in the previous section. By Theorem 5, the local search algorithm for MAX-SUM DIVERSIFICATION has approximation factor $1 - 2/k$. However, the running time of a straightforward application of Lemma 4 is $O(|\mathcal{X}| \cdot |E|\, k^2 \log k)$ even if the feasible solutions in $\mathcal{X}$ can be enumerated in $O(|\mathcal{X}| \cdot |E|)$ total time, which may be exponential in the input size $|E|$.

A main obstacle to applying the local search algorithm is that from a current set $\mathcal{Y} = \{Y_1, \ldots, Y_k\}$ of feasible solutions, we need to find a pair of feasible solutions $(X, Y) \in (\mathcal{X} \setminus \mathcal{Y}) \times \mathcal{Y}$ maximizing $d_{\text{sum}}(\mathcal{Y} - Y + X)$. To overcome this obstacle, we exploit *top-k enumeration algorithms*. Let $w' \colon E \to \mathbb{R}$ be a weight function. An algorithm $\mathcal{A}$ is called a *top-k enumeration algorithm for* $(E, \mathcal{X}, w', k)$ if for a positive integer k, $\mathcal{A}$ finds k feasible solutions $Y_1, \ldots, Y_k \in \mathcal{X}$ such that for any $Y \in \{Y_1, \ldots, Y_k\}$ and $X \in \mathcal{X} \setminus \{Y_1, \ldots, Y_k\}$, $w'(X) \le w'(Y)$ holds. By using $\mathcal{A}$, we can compute the pair (X, Y) as follows.

We first guess $Y \in \mathcal{Y}$ in the pair (X, Y) and let $\mathcal{Y}' = \mathcal{Y} \setminus \{Y\}$. To find the pair (X, Y), it suffices to find $X \in \mathcal{X} \setminus \mathcal{Y}'$ that maximizes $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y')$. For an element $e \in E$, we define a new weight $w'(e) := w(e)(\mathit{Ex}(e, \mathcal{Y}') - \mathit{In}(e, \mathcal{Y}'))$, where $\mathit{In}(e, \mathcal{Y}')$ (resp. $\mathit{Ex}(e, \mathcal{Y}')$) is the number of feasible solutions in $\mathcal{Y}'$ that contain e (resp. do not contain e). For notational convenience, we fix $\mathcal{Y}'$ and write $\mathit{In}(e)$ and $\mathit{Ex}(e)$ to denote $\mathit{In}(e, \mathcal{Y}')$ and $\mathit{Ex}(e, \mathcal{Y}')$, respectively. The following lemma shows that a feasible solution X that maximizes $w'(X)$ also maximizes $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y')$.

Lemma 7. *For any feasible solution $X \in \mathcal{X}$, $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y') = w'(X) + \sum_{e \in E} w(e) \cdot \mathit{In}(e)$.*

Proof. The contribution of $e \in X$ to $w(X \triangle Y')$ is $w(e)$ if $e \notin Y'$, and 0 otherwise. Thus, $e \in X$ contributes $w(e) \cdot \mathit{Ex}(e)$ to $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y')$. Similarly, $e \in E \setminus X$ contributes $w(e) \cdot \mathit{In}(e)$ to $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y')$. This gives us $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y') = w'(X) + \sum_{e \in E} w(e) \cdot \mathit{In}(e)$ as follows.

$$\begin{aligned}
&\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y') \\
&= \sum_{e \in X} w(e) \cdot \mathit{Ex}(e) + \sum_{e \in E \setminus X} w(e) \cdot \mathit{In}(e) \\
&= \sum_{e \in X} w(e)(\mathit{Ex}(e) - \mathit{In}(e)) + \sum_{e \in E} w(e) \cdot \mathit{In}(e) \\
&= w'(X) + \sum_{e \in E} w(e) \cdot \mathit{In}(e).
\end{aligned}$$

Notice that we use the following equation in the transformation of the equation in lines 2 and 3 $\sum_{e \in E \setminus X} w(e) \cdot \mathit{In}(e) = \sum_{e \in E} w(e) \cdot \mathit{In}(e) - \sum_{e \in X} w(e) \cdot \mathit{In}(e)$. □

From the above lemma, we can find the pair (X, Y) with a top-k enumeration algorithm $\mathcal{A}$ for $(E, \mathcal{X}, w', k)$ as follows. By Lemma 7, for any feasible solution $X \in \mathcal{X}$, $\sum_{Y' \in \mathcal{Y}'} w(X \triangle Y') = w'(X) + \sum_{e \in E} w(e) \cdot \mathit{In}(e)$. Since the second term does not depend on X, to find a feasible solution X maximizing the left-hand side, it suffices to maximize $w'(X)$ subject to $X \in \mathcal{X} \setminus \mathcal{Y}'$. The algorithm $\mathcal{A}$ allows us to find k feasible solutions $Z_1, \ldots, Z_k$ such that $w'(Z_1) \ge \cdots \ge w'(Z_k) \ge w'(Z)$ for any feasible solution Z other than $Z_1, \ldots, Z_k$. As $|\mathcal{Y}'| < k$, at least one of these solutions provides such a solution X.

The entire algorithm is as follows. We first find a set of k distinct feasible solutions in $\mathcal{X}$ using the enumeration algorithm $\mathcal{A}$. Then, we repeat the local update procedure described above $O(k \log k)$ times. Suppose that $\mathcal{A}$ enumerates k feasible solutions in time $O((|E| + k)^c)$ for some constant c. Then, the entire algorithm runs in time $O((|E|+k)^c |E| k^2 \log k)$ as we can compute the pair (X, Y) in time $O((|E| + k)^c |E| k)$ by simply guessing $Y \in \mathcal{Y}$.

The approximation factor $1 - 2/k$ does not give a reasonable bound for $k = 2$. In this case, however, we still have an approximation factor $1/2$ with a greedy algorithm for MAX-SUM DIVERSIFICATION [Birnbaum and Goldman 2009], which is described as follows. Initially, we set $\mathcal{Y} = \{Y_1\}$ with arbitrary $Y_1 \in \mathcal{X}$. Then, we compute a feasible solution $Y_2 \in \mathcal{X} \setminus \mathcal{Y}$ maximizing $\sum_{Y \in \mathcal{Y}} w(Y_2 \triangle Y)$. By Lemma 7 and the above discussion, we can find such a solution Y_2 with a top-k enumeration algorithm for $(E, \mathcal{X}, w', k)$, where $w'(e) := w(e) \cdot (\mathit{Ex}(e, \mathcal{Y}) - \mathit{In}(e, \mathcal{Y}))$ for $e \in E$. We repeat this $k - 1$ times so that $\mathcal{Y}$ contains k feasible solutions. As discussed in this section, the approximation factor of this algorithm is $1/2$ as in [Birnbaum and Goldman 2009]. Thus, the following theorem holds.

Theorem 8. *Let E be a finite set, $\mathcal{X} \subseteq 2^E$, and $w \colon E \to \mathbb{R}_{>0}$. Suppose that there is a top-k enumeration algorithm for $(E, \mathcal{X}, w', k)$ that runs in $O((|E| + k)^c)$ time, where $w' \colon E \to \mathbb{R}$ is an arbitrary weight function. Then, there is a $\max(1-2/k, 1/2)$-approximation algorithm for* MAX-SUM DIVERSE SOLUTIONS *that runs in $O((|E|+k)^c |E| k^2 \log k)$ time. Moreover, if there is a polynomial-time exact algorithm for* MAX-SUM DIVERSE SOLUTIONS *for constant k, then it admits a PTAS.*

We note that the approximation factor of the framework of Theorem 8 is gradually improved when the number k of solutions increases. However, the dependency of the running time on k is only polynomial, which allows us to find a moderate number of diverse solutions efficiently.

Applications of the Framework

To complete the description of approximation algorithms based on our framework, we need to develop top-k enumeration algorithms for specific problems. In this section, we design top-k enumeration algorithms for matchings, common bases of two matroids, and interval schedulings with cardinality r. Our top-k enumeration algorithms are based on a well-known technique used in [Lawler 1972] (also discussed in [Eppstein 2008]). The key to enumeration algorithms is the following WEIGHTED EXTENSION.

Definition 9 (WEIGHTED EXTENSION). *Given a finite set E, a set of feasible solutions $\mathcal{X} \subseteq 2^E$ as a membership oracle, a weight function $w' \colon E \to \mathbb{R}$, and a pair of disjoint subsets In and Ex of E, the task is to find a feasible solution $X \in \mathcal{X}$ that satisfies $In \subseteq X$ and $X \cap Ex = \emptyset$ maximizing $w'(X)$ subject to these conditions.*

If we can solve the above problem in $O(|E|^c)$ time, then we obtain a top-k enumeration algorithm for $(E, \mathcal{X}, w', k)$ that runs in $O(k\,|E|^{c+1})$ time.

Lemma 10 ([Lawler 1972]). *Suppose that* WEIGHTED EXTENSION *for $(E, \mathcal{X}, w', k)$ can be solved in $O(|E|^c)$ time. Then, there is an $O(k\,|E|^{c+1})$-time top-k enumeration algorithm for $(E, \mathcal{X}, w', k)$.*

Matchings

Matching is one of the most fundamental combinatorial objects in graphs, and the polynomial-time algorithm for computing a maximum weight matching due to [Edmonds 1965] is a cornerstone result in this context. Finding diverse matchings has also been studied so far [Hanaka et al. 2021, 2022; Fomin et al. 2020, 2021]. Let $G = (V, E)$ be a graph. A set of edges M is a *matching* of G if M has no pair of edges that share a common endpoint. A matching M is called a *perfect matching* of G if every vertex in G is incident to an edge in M. By using our framework, we design an approximation algorithm for finding diverse matchings. The formal definition of the problem is as follows.

Definition 11 (DIVERSE MATCHINGS). *Given a graph $G = (V, E)$, a weight function $w \colon E \to \mathbb{R}_{>0}$, and integers k and r, the task of* DIVERSE MATCHINGS *is to find k distinct matchings $M_1, \dots, M_k$ of cardinality at least r that maximize $d_{\mathrm{sum}}(\{M_1, \dots, M_k\})$.*

To apply our framework, it suffices to show that WEIGHTED EXTENSION for matchings can be solved in polynomial time. Our method is similar to a reduction from the maximum weight perfect matching problem to the maximum weight matching problem [Duan and Pettie 2014]. Let $In, Ex \subseteq E$ be disjoint subsets of edges and let $w' \colon E \to \mathbb{R}$. Then, our goal is to find a matching M of G with $|M| \geq r$ such that $In \subseteq M$ and $Ex \cap M = \emptyset$, and M maximizes $w'(M)$ subject to these constraints. By guessing the cardinality of M, it suffices to find such a matching M with cardinality exactly r. This problem can be reduced to that of finding a maximum weight perfect matching as follows. We assume that *In* is a matching of G as otherwise there is no matching containing it. Let $G' = (V', E')$ be the graph obtained from G by removing (1) all edges in *Ex* and (2) all end vertices of edges in *In*. Then, it is easy to see that M is a matching of G with $In \subseteq M$ and $Ex \cap M = \emptyset$ if and only if $M \setminus In$ is a matching of G'. Thus, it suffices to find a maximum weight matching of cardinality exactly $r' = r - |In|$ in G'. To this end, we add $|V'| - 2r'$ vertices U to G' and add all possible edges between vertices $v \in V'$ and $u \in U$. The graph obtained in this way is denoted by $H = (V' \cup U, E \cup F)$, where $F = \{\{u, v\} : u \in U, v \in V'\}$. We extend the weight function w' by setting $w'(f) = 0$ for $f \in F$. Then, the following lemma holds.

Lemma 12. *Let M^* be a maximum weight perfect matching in H. Then, $M^* \setminus F$ is a matching of cardinality r' in G' such that for every cardinality-r' matching M' in G', it holds that $w'(M') \leq w'(M^* \setminus F)$.*

Proof. Since M^* is a perfect matching and any edge incident to U is contained in F, M^* must contain exactly $|U|$ edges of F. This implies that the perfect matching M^* contains exactly r' edges of G'. Suppose that there is a cardinality-r' matching M' in G' such that $w'(M') > w'(M^* \setminus F)$. As every vertex in U is adjacent to V', we can choose exactly a set $N \subseteq F$ of $|U|$ edges between U and V' so that $M' \cup N$ forms a perfect matching in H. Then, we have $w'(M' \cup N) = w'(M') + w'(N) > w'(M^* \setminus F) + w'(M^* \cap F) = w'(M^*)$ as $w'(N) = w'(M^* \cap F) = 0$, contradicting the fact that M^* is a maximum weight perfect matching of H. $\square$

Thus, WEIGHTED EXTENSION for a matching of cardinality at least r is solvable in polynomial time using a maximum weight matching algorithm [Edmonds 1965]. By Theorem 8 and Lemma 10, we have the following theorem.

Theorem 13. *There is a polynomial-time approximation algorithm for* DIVERSE MATCHINGS *with approximation factor* $\max(1 - 2/k, 1/2)$.

Common Bases of Two Matroids

Let E be a finite set and let a non-empty family of subsets $\mathcal{I}$ of E. The pair $\mathcal{M} = (E, \mathcal{I})$ is a *matroid* if (1) for each $X \in \mathcal{I}$, every subset of X is included in $\mathcal{I}$ and (2) if $X, Y \in \mathcal{I}$ and $|X| < |Y|$, then there exists an element $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{I}$. Each set in $\mathcal{I}$ is called an *independent set* of $\mathcal{M}$. An inclusion-wise maximal independent set I of $\mathcal{M}$ is a *base* of $\mathcal{M}$. Because of condition (2), all bases in $\mathcal{M}$ have the same cardinality. For two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$, a subset $X \subseteq E$ is a *common base of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ if X is a base of both $\mathcal{M}_1$ and $\mathcal{M}_2$. In this subsection, we give an approximation algorithm for diverse common bases of two matroids.

Definition 14 (DIVERSE MATROID COMMON BASES). *Given two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ as membership oracles, a weight function $w \colon E \to \mathbb{R}_{>0}$, and an integer k, the task of* DIVERSE MATROID COMMON BASES *is to find k distinct common bases $B_1, \dots, B_k$ of $\mathcal{M}_1$ and $\mathcal{M}_2$ that maximize $d_{\mathrm{sum}}(B_1, \dots, B_k)$.*

Given two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ as membership oracles, the problem of partitioning E into k common bases of $\mathcal{M}_1$ and $\mathcal{M}_2$ is a notoriously hard problem, which requires an exponential number of membership queries [Bérczi and Schwarcz 2021]. This fact together with Observation 2 implies that DIVERSE MATROID COMMON BASES cannot be solved with polynomial number of membership queries in our problem setting. Given this fact, we develop a constant-factor approximation algorithm for DIVERSE MATROID COMMON BASES. To this end, we show that WEIGHTED EXTENSION for common bases of two matroids can be solved in polynomial time.

Similarly to the case of matchings, we can find a maximum weight common base $B \in \mathcal{I}_1 \cap \mathcal{I}_2$ subject to $In \subseteq B$ and $Ex \cap B = \emptyset$ for given disjoint $In, Ex \subseteq E$, which is as follows. Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. For $X \subseteq E$, we let $\mathcal{M} \setminus X = (E \setminus X, \mathcal{J})$, where $\mathcal{J} = \{J \setminus X : J \in \mathcal{I}\}$. Then, $\mathcal{M} \setminus X$ is a matroid (see [Oxley 2006]). Similarly, for $X \subseteq E$, we let $\mathcal{M}/X = (E \setminus X, \mathcal{J}')$, where $\mathcal{J}' = \{J : J \cup X \in \mathcal{I}, J \subseteq E \setminus X\}$. Then $(E, \mathcal{J})$ is also a matroid (see [Oxley 2006]). For two matroids $\mathcal{M}_1$ and $\mathcal{M}_2$, we consider two matroids $\mathcal{M}_1' = (\mathcal{M}_1 \setminus Ex)/In$ and $\mathcal{M}_2' = (\mathcal{M}_2 \setminus Ex)/In$. For every independent set X in $\mathcal{M}_1'$ and $\mathcal{M}_2'$, X does not contain any element in Ex and $X \cup In$ is an independent set in both $\mathcal{M}_1$ and $\mathcal{M}_2$. Thus, WEIGHTED EXTENSION can be solved by computing a maximum weight common base in $\mathcal{M}_1'$ and $\mathcal{M}_2'$, which can be solved in polynomial time (see Theorem 41.7 in [Schrijver 2003]). By Theorem 8 and Lemma 10, the following theorem holds.

Theorem 15. *There is a polynomial-time approximation algorithm for* DIVERSE MATROID COMMON BASES *with approximation factor* $\max(1 - 2/k, 1/2)$*, provided that the membership oracles for* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *can be evaluated in polynomial time.*

Minimum Cuts

Let $G = (V, E)$ be a graph. A partition of V into two nonempty sets A and B is called a *cut* of G. For a cut (A, B) of G, the set of edges having one end in A and the other end in B is denoted by $E(A, B)$. When no confusion arises, we may refer to $E(A, B)$ as a cut of G. The *size* of a cut $C = E(A, B)$ is defined by $|E(A, B)|$. A cut C is called a *minimum cut* of G if there is no cut C' of G with $|C'| < |C|$. In this section, we consider the following problem.

Definition 16 (DIVERSE MINIMUM CUTS). *Given a graph* $G = (V, E)$ *with an edge-weight function* $w\colon E \to \mathbb{R}_{\geq 0}$ *and an integer* k*, the task of* DIVERSE MINIMUM CUTS *is to find* k *distinct minimum cuts* $C_1, \ldots, C_k \subseteq E$ *of* G *that maximize* $d_{\mathrm{sum}}(\{C_1, \ldots, C_k\})$.

An important observation for this problem is that the number of minimum cuts of any graph G is $O(|V|^2)$ [Karger 2000]. Moreover, we can enumerate all minimum cuts in a graph in polynomial time [Yeh, Wang, and Su 2010; Vazirani and Yannakakis 1992]. Thus, we can solve both WEIGHTED EXTENSION for minimum cuts and DIVERSE MINIMUM CUTS for constant k in polynomial time, yielding a PTAS for DIVERSE MINIMUM CUTS.

Theorem 17. DIVERSE MINIMUM CUTS *admits a PTAS.*

Given this, it is natural to ask whether DIVERSE MINIMUM CUTS admits a polynomial-time algorithm. However, we show that DIVERSE MINIMUM CUTS is NP-hard even if G has a cut of size 3. Let $\lambda(G)$ be the size of a minimum cut of G.

Theorem 18 ($\star$). DIVERSE MINIMUM CUTS *is* NP*-hard even if* $\lambda(G) = 3$.

The proof of the theorem goes as follows. Suppose that $\lambda(G) = 3$. Suppose moreover that G has no nontrivial cuts of size 3 (i.e., $|E(A, B)| > 3$ for any $A \subseteq V$ with $\min(|A|, |B|) \geq 2$). Then, every minimum cut of G separates a vertex from the other vertices. If two minimum cuts of G do not share an edge, the corresponding two vertices have to be nonadjacent. Thus, G has k edge-disjoint minimum cuts if and only if G has an independent set of size k. Our reduction proves this correspondence in a similar argument without the assumption on nontrivial cuts.

When $\lambda(G) = 1$, then DIVERSE MINIMUM CUTS is trivially solvable in linear time as the problem can be reduced to finding all bridges in G. If $\lambda(G) = 2$, the problem is slightly nontrivial, which in fact is solvable in polynomial time as well.

Theorem 19 ($\star$). DIVERSE MINIMUM CUTS *can be solved in* $|V|^{O(1)}$ *time, provided that* $\lambda(G) \leq 2$.

We reduce the problem to that of finding a subgraph of prescribed size with maximizing the sum of convex functions on their degrees of vertices, which can be solved in polynomial time [Apollonio and Sebö 2009].

Interval Schedulings

For a pair of integers a and b with $a \leq b$, the set of all numbers between a and b is denoted by $[a, b]$. We call $I = [a, b]$ an *interval*. For a pair of intervals $I = [a, b]$ and $J = [c, d]$, we say that I *overlaps* J if $I \cap J \neq \emptyset$. For a set of intervals $\mathcal{S} = \{I_1, \ldots, I_r\}$, we say that $\mathcal{S}$ is a *valid scheduling* (or simply a *scheduling*) if for any pair of intervals $I_i, I_j \in \mathcal{S}$, I_i does not overlap I_j. In particular, we call $\mathcal{S}$ an r*-scheduling* if $|\mathcal{S}| = r$ for $r \in \mathbb{N}$. In this section, we deal with the following problem.

Definition 20 (DIVERSE INTERVAL SCHEDULINGS). *Given a set of intervals* $\mathcal{I} = \{I_1, \ldots, I_n\}$*, a weight function* $w\colon \mathcal{I} \to \mathbb{R}_{>0}$*, and integers* k *and* r*, the task of* DIVERSE INTERVAL SCHEDULINGS *is to find* k *distinct* r*-schedulings* $\mathcal{S}_1, \ldots, \mathcal{S}_k \subseteq \mathcal{I}$ *that maximize* $d_{\mathrm{sum}}(\{\mathcal{S}_1, \ldots, \mathcal{S}_k\})$.

Since the problem of partitioning a set of intervals $\mathcal{I}$ into k scheduling $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that each $\mathcal{S}_i$ has exactly r intervals is NP-hard [Bodlaender and Jansen 1995; Gardi 2009][2], by Observation 2, the following theorem holds.

Theorem 21. DIVERSE INTERVAL SCHEDULINGS *is* NP*-hard.*

To apply Theorem 8 to DIVERSE INTERVAL SCHEDULINGS, it suffices to give a polynomial-time algorithm for WEIGHTED EXTENSION for interval schedulings. Observe that if In is not a scheduling, then there is no scheduling containing In. Observe also that we can remove all intervals included in Ex or overlapping some interval in In. Thus, the problem can be reduced to the one for finding a maximum weight scheduling with cardinality $r' = r - |In|$. This problem can be solved in polynomial time by using a simple dynamic programming approach.

[2]Note that the NP-hardness is proven for the case that each $\mathcal{S}_i$ has *at most* r intervals, but a simple reduction proves the NP-hardness of this variant.

Lemma 22. *Given a set $\mathcal{I}$ and $w' : \mathcal{I} \to \mathbb{R}$ and $r' \in \mathbb{N}$, there is a polynomial-time algorithm finding a maximum weight r'-scheduling in $O(|\mathcal{I}|^2 r')$ time.*

Proof. The algorithm is analogous to that to find a maximum weight independent set on interval graphs, which is roughly sketched as follows. We assume that $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ is sorted with respect to their right end points. We define $\mathrm{opt}(p, q)$ as the maximum total weight of a q-scheduling $\mathcal{S}$ in $\{I_1, I_2, \ldots, I_p\}$ such that $I_p \in \mathcal{S}$ for $0 \le p \le n$ and $0 \le q \le r'$. Then, the values of $\mathrm{opt}(p, q)$ for all p and q can be computed by a standard dynamic programming algorithm in time $O(|\mathcal{I}|^2 r')$. $\square$

By Theorem 8 and Lemma 10, we obtain a polynomial-time approximation algorithm for DIVERSE INTERVAL SCHEDULINGS with factor $\max(1 - 2/k, 1/2)$.

Finally, we show that DIVERSE INTERVAL SCHEDULINGS can be solved in polynomial time for fixed k using a dynamic programming approach, which implies a PTAS for DIVERSE INTERVAL SCHEDULINGS.

Similarly to the proof of Lemma 22, assume that $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ is sorted with respect to their right end points. Let $[k] = \{1, 2, \ldots, k\}$. For each $0 \le p \le |\mathcal{I}|$, we consider a tuple $T = (p, L, R, \Gamma)$, where L and R are vectors in $([n] \cup \{0\})^k$ and $([r] \cup \{0\})^k$, respectively, and Γ is a subset of $\binom{[k]}{2}$. Clearly, the number of tuples is $O(n(n+1)^k (r+1)^k 2^{\binom{k}{2}})$, which is polynomial when k is a constant. We denote by ℓ_i and r_i the ith component of L and R, respectively. For a tuple $T = (p, L, R, \Gamma)$, the value $\mathrm{opt}(T)$ is the maximum value of $d_{\mathrm{sum}}(\{\mathcal{S}_1, \ldots, \mathcal{S}_k\})$ for k schedulings under the following four conditions: (1) the maximum index of an interval in $\bigcup_{1 \le i \le k} \mathcal{S}_i$ is p ($p = 0$ if $\bigcup_{1 \le i \le k} \mathcal{S}_i = \emptyset$); (2) for $1 \le i \le k$, the maximum index of an interval in $\mathcal{S}_i$ is ℓ_i ($\ell_i = 0$ if $\mathcal{S}_i = \emptyset$); (3) for $1 \le i \le k$, $|\mathcal{S}_i| = r_i$; and (4) for $1 \le i < j \le k$, $\{i, j\} \in \Gamma$ if and only if $\mathcal{S}_i$ and $\mathcal{S}_j$ are distinct.

We define $\mathrm{opt}(T) = -\infty$ if no such a set of scheduings exists. When $R = (r, r, \ldots, r)$ and $\Gamma = \binom{[k]}{2}$, there is a set of k distinct r-schedulings that have the sum diversity $\mathrm{opt}(T)$ unless $\mathrm{opt}(T) = -\infty$. For a tuple T, we say that a set of k schedulings is *valid for* T if it satisfies the above four conditions. Hence, among the tuples of the form (p, L, R, Γ) with $R = (r, \ldots, r)$ and $\Gamma = \binom{[k]}{2}$, $\mathrm{opt}(T)$ is the optimal value for DIVERSE INTERVAL SCHEDULINGS. We next explain the outline of our dynamic programming algorithm to compute $\mathrm{opt}(T)$ for any T.

As a base case, $p = 0$, $L = (0, \ldots, 0)$, $R = (0, \ldots, 0)$, and $\Gamma = \emptyset$ if and only if $\mathrm{opt}(T) = 0$. Let T' be a tuple (p', L', R', Γ') that satisfies the following conditions: (1)$p' < p$; (2) for any $1 \le i \le k$, $\ell'_i \le \ell_i$ and $r'_i \le r_i$; and (3) $\Gamma' \subseteq \Gamma$. We say that a tuple T' satisfying the above conditions is *dominated by* T. We denote the set of tuples dominated by T as $D(T)$. Let $C(T) = \{i : \ell_i = p\}$. A tuple T' is *valid for* T if T' satisfies the following conditions: (1) $T' \in D(T)$; (2) if $i \in C(T)$ and $\ell_i > 0$, then interval I_{ℓ_i} does not overlap with I_p; (3) if $i \in C(T)$, $r'_i = r_i - 1$, otherwise, $r'_i = r_i$; and (4) $\Gamma = \Gamma' \cup P(T)$ with $P(T) := \{\{i, j\} \in \binom{[k]}{2} : |\{i, j\} \cap C(T)| = 1\}$. We denote the set of valid tuples for T as $V(T)$. We compute $\mathrm{opt}(T)$ using the following lemma.

Lemma 23. *For a tuple T, $\mathrm{opt}(T)$ equals the following formula.*

$$\max_{T' \in V(T)} (\mathrm{opt}(T') + w(I_p) \cdot |C(T)| \cdot (k - |C(T)|)).$$

Proof. Let $T = (p, L, R, \Gamma)$. Let $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ be a valid set of schedulings with $d_{\mathrm{sum}}(\{\mathcal{S}_1, \ldots, \mathcal{S}_j\}) = \mathrm{opt}(T)$. Then, $\mathcal{S}' = (\mathcal{S}_1 \setminus \{I_p\}, \ldots, \mathcal{S}_k \setminus \{I_p\})$ is a valid set of scheduings for $T' \in V(T)$. Moreover, $d_{\mathrm{sum}}(\mathcal{S}) = d_{\mathrm{sum}}(\mathcal{S}') + w(I_p) \cdot |C(T)| \cdot (k - |C(T)|)$ as I_p contributes $w(I_p) \cdot |C(T)| \cdot (k - |C(T)|)$ to the diversity. Thus, the left-hand side is at most the right-hand side.

Conversely, let T' be a tuple maximizing the left-hand side and let $\mathcal{S}' = \{\mathcal{S}'_1, \ldots, \mathcal{S}'_k\}$ be a valid set of schedulings for T'. For each $1 \le i \le k$, we set $\mathcal{S}_i = \mathcal{S}'_i \cup \{I_p\}$ if $i \in C(T)$ and $\mathcal{S}_i = \mathcal{S}'_i$ otherwise. By condition (2) in the definition of a valid tuple, each interval in $\mathcal{S}'_i$ does not overlap with I_p, meaning that $\mathcal{S}_i$ is a scheduling. Thus, the right-hand side is at most the left-hand side. $\square$

Thus, we can compute $\mathrm{opt}(T)$ for any T in polynomial time when k is a constant. Moreover, from $\mathrm{opt}(T)$, we can find k schedulings with the maximum sum diversity by a standard trace back technique. Combining the approximation algorithm and the above algorithm, we obtain a PTAS.

Theorem 24. DIVERSE INTERVAL SCHEDULINGS *admits a PTAS.*

It is not hard to see that the above algorithm is modified into the one finding a set $\mathcal{S}$ of k valid schedulings $\mathcal{S}_i$ with $|\mathcal{S}_i| \ge r$ maximizing $d_{\mathrm{sum}}(\{\mathcal{S}_1, \ldots, \mathcal{S}_k\})$. The modified algorithm simply takes "at least r intervals" instead of "exactly r intervals", which runs in polynomial time as well.

Conclusion

We give a framework for designing approximation algorithms for MAX-SUM DIVERSE SOLUTIONS. This framework runs in $\mathrm{poly}(|E| + k)$ time and is versatile, which allows to apply to the diverse version of several well-studied combinatorial problems. The key to applying our framework is a polynomial-time algorithm for WEIGHTED EXTENSION, which yields constant-factor approximation algorithms for DIVERSE MATCHINGS and DIVERSE MATROID COMMON BASES. Moreover, we obtain a PTAS for MAX-SUM DIVERSE SOLUTIONS if we can solve the problem in polynomial time for fixed k, yielding PTASes for DIVERSE MINIMUM CUTS and DIVERSE INTERVAL SCHEDULINGS.

There are several directions from our work. Our approximation algorithms for DIVERSE MATCHINGS and DIVERSE MATROID COMMON BASES give a approximation factor $\max(1-2/k, 1/2)$, which is a constant when k is a constant. The APX-hardness of these problems is an interesting question to prove a limitation of finding "approximately" diverse solutions. Our work focuses only on MAX-SUM HAMMING DISTANCE as our objective function. However, MAX-MIN HAMMING DISTANCE or other diversity measures would be more acceptable in some practical applications. It would be worth investigating these diversity measures from the viewpoint of approximability.

Acknowledgements

The authors thank Jaehoon Yu and Shungo Kumazawa for helpful discussions. This work was partially supported by JSPS Kakenhi Grant Numbers JP18H04091, JP18K11169, JP20H05793, JP20H05795, JP20K11692, JP20K19742, JP21K11752, JP21K17812, JP21H05861, and JP21H05852, JP21K17707, JP22H00513, JST CREST Grant Number JPMJCR18K3, and JST ACT-X Grant Number JPMJAX2105, Japan.

References

Apollonio, N.; and Sebö, A. 2009. Minconvex Factors of Prescribed Size in Graphs. *SIAM J. Discret. Math.*, 23(3): 1297–1310.

Baste, J.; Fellows, M. R.; Jaffke, L.; Masařík, T.; de Oliveira Oliveira, M.; Philip, G.; and Rosamond, F. A. 2022. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303: 103644.

Baste, J.; Jaffke, L.; Masarík, T.; Philip, G.; and Rote, G. 2019. FPT Algorithms for Diverse Collections of Hitting Sets. *Algorithms*, 12(12): 254.

Bérczi, K.; and Schwarcz, T. 2021. Complexity of packing common bases in matroids. *Math. Program.*, 188(1): 1–18.

Birnbaum, B. E.; and Goldman, K. J. 2009. An Improved Analysis for a Greedy Remote-Clique Algorithm Using Factor-Revealing LPs. *Algorithmica*, 55(1): 42–59.

Bodlaender, H. L.; and Jansen, K. 1995. Restrictions of Graph Partition Problems. Part I. *Theor. Comput. Sci.*, 148(1): 93–109.

Cevallos, A.; Eisenbrand, F.; and Zenklusen, R. 2016. Max-Sum Diversity Via Convex Programming. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 26:1–26:14. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-009-5.

Cevallos, A.; Eisenbrand, F.; and Zenklusen, R. 2019. An Improved Analysis of Local Search for Max-Sum Diversification. *Math. Oper. Res.*, 44(4): 1494–1509.

Danna, E.; Fenelon, M.; Gu, Z.; and Wunderling, R. 2007. Generating Multiple Solutions for Mixed Integer Programming Problems. In Fischetti, M.; and Williamson, D. P., eds., *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, volume 4513 of *Lecture Notes in Computer Science*, 280–294. Springer.

Danna, E.; and Woodruff, D. L. 2009. How to Select a Small Set of Diverse Solutions to Mixed Integer Programming Problems. *Oper. Res. Lett.*, 37(4): 255–260.

Deza, M. M.; and Laurent, M. 1997. *Geometry of cuts and metrics*, volume 2. Springer.

Drosou, M.; and Pitoura, E. 2010. Search Result Diversification. *SIGMOD Rec.*, 39(1): 41–47.

Duan, R.; and Pettie, S. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM*, 61(1).

Edmonds, J. 1965. Paths, Trees, and Flowers. *Canadian J. Math.*, 17: 449–467.

Eppstein, D. 2008. k-Best Enumeration. In *Encyclopedia of Algorithms*, 1–4. Boston, MA: Springer US. ISBN 978-3-642-27848-8.

Fernau, H.; Golovach, P.; Sagot, M.-F.; et al. 2019. Algorithmic enumeration: Output-sensitive, input-sensitive, parameterized, approximative (Dagstuhl Seminar 18421). In *Dagstuhl Reports*, volume 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Fomin, F. V.; Golovach, P. A.; Jaffke, L.; Philip, G.; and Sagunov, D. 2020. Diverse Pairs of Matchings. In *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Fomin, F. V.; Golovach, P. A.; Panolan, F.; Philip, G.; and Saurabh, S. 2021. Diverse Collections in Matroids and Graphs. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Gao, J.; Goswami, M.; Karthik C. S.; Tsai, M.; Tsai, S.; and Yang, H. 2022. Obtaining Approximately Optimal and Diverse Solutions via Dispersion. In Castañeda, A.; and Rodríguez-Henríquez, F., eds., *LATIN 2022: Theoretical Informatics - 15th Latin American Symposium, Guanajuato, Mexico, November 7-11, 2022, Proceedings*, volume 13568 of *Lecture Notes in Computer Science*, 222–239. Springer.

Gardi, F. 2009. Mutual exclusion scheduling with interval graphs or related classes, Part I. *Discret. Appl. Math.*, 157(1): 19–35.

Hanaka, T.; Kobayashi, Y.; Kurita, K.; Lee, S. W.; and Otachi, Y. 2022. Computing Diverse Shortest Paths Efficiently: A Theoretical and Experimental Study. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 3758–3766.

Hanaka, T.; Kobayashi, Y.; Kurita, K.; and Otachi, Y. 2021. Finding Diverse Trees, Paths, and More. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 3778–3786.

Hao, F.; Pei, Z.; and Yang, L. T. 2020. Diversified top-k maximal clique detection in Social Internet of Things. *Future Generation Computer Systems*, 107: 408–417.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding Diverse and Similar Solutions in Constraint Programming. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, 372–377. AAAI Press. ISBN 157735236x.

Hentenryck, P. V.; Coffrin, C.; and Gutkovich, B. 2009. Constraint-Based Local Search for the Automatic Generation of Architectural Tests. In Gent, I. P., ed., *Principles and*

Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings, volume 5732 of *Lecture Notes in Computer Science*, 787–801. Springer.

Holyer, I. 1981. The NP-Completeness of Edge-Coloring. *SIAM J. Comput.*, 10(4): 718–720.

Ingmar, L.; Garcia de la Banda, M.; Stuckey, P. J.; and Tack, G. 2020. Modelling Diversity of Solutions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02): 1528–1535.

Karger, D. R. 2000. Minimum Cuts in Near-Linear Time. *J. ACM*, 47(1): 46–76.

Lawler, E. L. 1972. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7): 401–405.

Nadel, A. 2011. Generating Diverse Solutions in SAT. In Sakallah, K. A.; and Simon, L., eds., *Theory and Applications of Satisfiability Testing - SAT 2011*, 287–301. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-21581-0.

Oxley, J. G. 2006. *Matroid Theory (Oxford Graduate Texts in Mathematics)*. USA: Oxford University Press, Inc. ISBN 0199202508.

Petit, T.; and Trapp, A. C. 2015. Finding Diverse Solutions of High Quality to Constraint Optimization Problems. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 260–267.

Petit, T.; and Trapp, A. C. 2019. Enriching Solutions to Combinatorial Problems via Solution Engineering. *INFORMS Journal on Computing*, 31(3): 429–444.

Ravi, S. S.; Rosenkrantz, D. J.; and Tayi, G. K. 1994. Heuristic and Special Case Algorithms for Dispersion Problems. *Operations Research*, 42(2): 299–310.

Schrijver, A. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

Vazirani, V. V.; and Yannakakis, M. 1992. Suboptimal cuts: Their enumeration, weight and number. In Kuich, W., ed., *Automata, Languages and Programming*, 366–377. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-47278-0.

Vieira, M. R.; Razente, H. L.; Barioni, M. C. N.; Hadjieleftheriou, M.; Srivastava, D.; Jr., C. T.; and Tsotras, V. J. 2011. On query result diversification. In Abiteboul, S.; Böhm, K.; Koch, C.; and Tan, K., eds., *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, 1163–1174. IEEE Computer Society.

Wang, J.; Cheng, J.; and Fu, A. W. 2013. Redundancy-aware maximal cliques. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, 122–130. ACM.

Yeh, L.-P.; Wang, B.-F.; and Su, H.-H. 2010. Efficient Algorithms for the Problems of Enumerating Cuts by Non-Decreasing Weights. *Algorithmica*, 56(3): 297–312.

Yuan, L.; Qin, L.; Lin, X.; Chang, L.; and Zhang, W. 2016. Diversified top-k clique search. *VLDB J.*, 25(2): 171–196.