Obtaining Approximately Optimal and Diverse Solutions via Dispersion

Jie Gao ✉ ⬤
Department of Computer Science, Rutgers University, USA

Mayank Goswami ✉ ⬤
Department of Computer Science, Queens College, City University of New York, USA

Karthik C. S. ✉
Department of Computer Science, Rutgers University, USA

Meng-Tsung Tsai ✉ ⬤
Institute of Information Science, Academia Sinica, Taiwan

Shih-Yu Tsai ✉
Department of Computer Science, Stony Brook University, USA

Hao-Tsung Yang ✉
School of Informatics, University of Edinburgh, UK

──── **Abstract** ────

There has been a long-standing interest in computing diverse solutions to optimization problems. Motivated by reallocation of governmental institutions in Sweden, in 1995 J. Krarup [26] posed the problem of finding k edge-disjoint Hamiltonian Circuits of minimum total weight, called the peripatetic salesman problem (PSP). Since then researchers have investigated the complexity of finding diverse solutions to spanning trees, paths, vertex covers, matchings, and more. Unlike the PSP that has a constraint on the total weight of the solutions, recent work has involved finding diverse solutions that are all optimal.

However, sometimes the space of exact solutions may be too small to achieve sufficient diversity. Motivated by this, we initiate the study of obtaining sufficiently-diverse, yet approximately-optimal solutions to optimization problems. Formally, given an integer k, an approximation factor c, and an instance I of an optimization problem, we aim to obtain a set of k solutions to I that a) are all c approximately-optimal for I and b) maximize the diversity of the k solutions. Finding such solutions, therefore, requires a better understanding of the global landscape of the optimization function.

We show that, given any metric on the space of solutions, and the diversity measure as the sum of pairwise distances between solutions, this problem can be solved by combining ideas from dispersion and multicriteria optimization. We first provide a general reduction to an associated budget-constrained optimization (BCO) problem, where one objective function is to be maximized (minimized) subject to a bound on the second objective function. We then prove that bi-approximations to the BCO can be used to give bi-approximations to the diverse approximately optimal solutions problem with a little overhead. We then give the following applications of our result:

- Polynomial time $(2, c)$ bi-approximation algorithms for diverse c-approximate maximum matchings, and for diverse c-approximate st shortest paths.
- Polynomial time $(4, 2c)$ bi-approximation, a PTAS $(4, (1 + \epsilon)c)$ bi-approximation and pseudopolynomial $(4, c)$ bi-approximation algorithms for diverse c-approximate minimum weight bases of a matroid. In particular, this gives us diverse c-approximate minimum spanning trees, advancing a step towards achieving diverse c-approximate TSP tours.

We explore the connection to the field of multiobjective optimization and show that the class of problems to which our result applies includes those for which the associated DUALRESTRICT problem defined by Papadimitriou and Yannakakis [35], and recently explored by Herzel et al. [24] can be solved in polynomial time.

On the way we prove a conditional 2-hardness of approximation lower bound on the problem of k-dispersion in metric spaces, resolving (up to ETH) the optimality of the greedy farthest point

heuristic for dispersion. This also provides evidence of tightness of the factor 2 for diversity in most of our algorithms.

1   Introduction

Techniques for optimization problems focus on obtaining optimal solutions to an objective function and have widespread applications ranging from machine learning, operations research, computational biology, networks, to geophysics, economics, and finance. However, in many scenarios, the optimal solution is not only computationally difficult to obtain, but can also render the system built upon its utilization vulnerable to adversarial attacks. Consider a patrolling agent tasked with monitoring n sites in the plane. The most efficient solution (i.e., maximizing the frequency of visiting each of the n sites) would naturally be to patrol along the tour of shortest length[1] (the solution to TSP - the Traveling Salesman Problem). However, an adversary who wants to avoid the patroller can also compute the shortest TSP tour and can design its actions strategically [39]. Similarly, applications utilizing the minimum spanning tree (MST) on a communication network may be affected if an adversary gains knowledge of the network [11]; systems using solutions to a linear program (LP) would be vulnerable if an adversary gains knowledge of the program's function and constraints.

One way to address the vulnerability is to use a set of approximately optimal solutions and randomize among them. However, this may not help much to mitigate the problem, if these approximate solutions are combinatorially too "similar" to the optimal solution. For example, all points in a sufficiently small neighborhood of the optimal solution on the LP polytope will be approximately optimal, but these solutions are not too much different and the adversaries can still effectively carry out their attacks. Similarly one may use another tree instead of the MST, but if the new tree shares many edges with the MST the same vulnerability persists. Thus k-best enumeration algorithms ([17, 22, 28, 29, 33]) developed for a variety of problems fall short in this regard.

One of the oldest known formulations is the Peripatetic Salesman problem (PSP) by Krarup [26], which asks for k edge-disjoint Hamiltonian circuits of minimum total weight in a network. Since then, several researchers have tried to compute diverse solutions for several optimization problems [3, 4, 15, 21]. Most of these works are on graph problems,

---

[1] We assume without loss of generality that the optimal TSP is combinatorially unique by a slight perturbation of the distances.

and diversity usually corresponds to the size of the symmetric difference of the edge sets in the solutions. Crucially, almost all of the aforementioned work demands either every solution individually be optimal, or the set of solutions in totality (as in the case of the PSP) be optimal. Nevertheless, **the space of optimal solutions may be too small to achieve sufficient diversity**, and it may just be singular (unique solution). In addition, for NP-complete problems finding just one optimal solution is already difficult. While there is some research that takes the route of developing FPT algorithms for this setting [4, 16], to us it seems practical to also consider the relaxation to approximately-optimal solutions.

This motivates the problem of finding a set of diverse and *approximately optimal* solutions, which is the problem considered in this article. The number of solutions k and the desired approximation factor $c > 1$ is provided by the user as input. Working in the larger class gives one more hope of finding diverse solutions, yet every solution has a guarantee on its quality.

1.1 Our Contributions

We develop approximation algorithms for finding k solutions to the given optimization problem: for every solution, the quality is bounded by a user-given approximation ratio $c > 1$ to the optimal solution and the diversity of these k solutions is maximized. Given a metric on the space of solutions to the problem, we consider the diversity measure given by the sum (or average) of pairwise distances between the k solutions. Combining ideas from the well-studied problem on dispersion (which we describe next), we reduce the above problem to a budget constrained optimization (BCO) program.

1.2 Dispersion

Generally speaking, if the optimization problem itself is $\mathcal{NP}$-hard, finding diverse solutions for that problem is also $\mathcal{NP}$-hard (see Proposition 5 for more detail). On the other hand, interestingly, even if the original problem is not $\mathcal{NP}$-hard, finding diverse and approximately optimal solutions can still be $\mathcal{NP}$-hard. This is due to the connection of the diversity maximization objective with the general family of problems that consider selecting k elements from the given input set with maximum "dispersion", defined as max-min distance, max-average distance, and so on.

The dispersion problem has a long history, with many variants both in the metric setting and the geometric setting [14, 27, 38]. For example, finding a subset of size k from an input set of n points in a metric space that maximizes the distance between closest pairs or the sum of distances of the k selected points are both $\mathcal{NP}$-hard [1, 37]. For the max-sum dispersion problem, the best known approximation factor is 2 in the metric setting [6, 23] and $1.571 + \epsilon$ with $\epsilon > 0$ for the 2D Euclidean setting [37]. It is not known whether there is a PTAS for the max-min dispersion problem.

Conditional hardness In this paper, we provide new hardness results for the max-sum k-dispersion problem. We show that, assuming a standard complexity theoretic conjecture (Exponential Time Hypothesis), there is no polynomial time algorithm for the k-dispersion problem in a metric space which approximates it to a factor better than 2 (Theorem 9). This provides a tight understanding for the max-sum k-dispersion problem which has been studied since the 90s [37].

Dispersion in exponentially-sized space We make use of the general framework of the 2-approximation algorithm [7, 37] to the max-sum k-dispersion problem, a greedy algorithm where the jth solution is chosen to be the most distant/diverse one from the first $j - 1$ solutions. Notice that in our setting, there is an important additional challenge to understand

the space within which the approximate solutions stay. In all of the problems we study, the total number of solutions can be *exponential in the input size.* Thus we need to have a non-trivial way of navigating within this large space and carry furthest insertion without considering all points in the space. This is where our reduction to budget constrained problem comes in.

Self avoiding dispersion Furthermore, even after implicitly defining the jth furthest point insertion via some optimization problem, one needs to take care that the (farthest, in terms of sum of distances) solution does not turn out to equal one of the previously found $j - 1$ solutions, as this is a requirement for the furthest point insertion algorithm. This is an issue one faces because of the implicit nature of the furthest point procedure in the exponential-sized space of solutions: in the metric k-dispersion problem, it was easy to guarantee distinctness as one only considered the $n - (j - 1)$ points not yet selected.

1.3   Reduction to Budget Constrained Optimization

Combining with dispersion, we reduce the diversity computational problem to a budget constrained optimization (BCO) problem where the budget is an upper (resp. lower) bound if the quality of solution is described by a minimization (resp. maximization) problem. Recall that the number of solutions k and the approximation factor c is input by the user.

We show how using an (a, b) bi-approximation algorithm for the BCO problem provides a set of $O(a)$-diverse, bc approximately-optimal solutions to the diversity computational problem (the hidden constant is at most 4). This main reduction is described in Theorem 13.

The main challenge in transferring the bi-approximation results because of a technicality that we describe next. Let $\Omega(c)$ be the space of c approximate solutions. A $(*, b)$ bi-approximation algorithm to the BCO relaxes the budget constraint by a factor b, and hence only promises to return a faraway point in the larger space $\Omega(b \cdot c)$. Thus bi-approximation of BCO do not simply give a farthest point insertion in the space of solutions, but return a point in a larger space. Nevertheless, we prove in Lemma 15 that in most cases, one loses a factor of at most 4 in the approximation factor for the diversity.

Once the reduction to BCOs is complete, for diverse approximate matchings, spanning trees and shortest paths we exploit the special characteristics of the corresponding BCO to solve it optimally ($a = b = 1$). For other problems such as diverse approximate minimum weight spanning trees, and the more general minimum weight bases of a matroid, we utilize known bi-approximations to the BCO to obtain bi-approximations for the diversity problem. For all problems except diverse (unweighted) spanning trees[2], our algorithms are the first polynomial time bi-approximations for these problems.

We also connect to the wide literature on multicriteria optimization and show that our result applies to the entire class of problems for which the associated DUALRESTRICT problem (defined by Papadimitriou and Yannakakis [35], and recently studied by Herzel et al. [24]) has a polynomial time solution. We discuss this in more detail after presenting our reduction.

The rest of this paper is organized as follows: we survey related work in Section 2, and formulate the problem in Section 3. In Section 4 we describe our main results on dispersion (Theorem 9) and the reduction to the budget constrained optimization problem (Theorem 13).

---

[2] While an exact algorithm for diverse unweighted spanning trees is known [21], we give a faster (by a factor $\Omega(n^{1.5}k^{1.5}/\alpha(n, m))$ where $\alpha(\cdot)$ denotes the inverse of the Ackermann function) 2-approximation here.

Sections 5–8 describe four applications of our technique to various problems such as diverse approximate matchings, shortest paths, minimum spanning trees, and minimum weight bases of a matroid. We remark that this list is by no means exhaustive, and we leave finding other interesting optimization problems which are amenable to our approach for future research. We end with open problems in Section 9.

2   Related Work

Recently there has been a surge of interest in the tractability of finding diverse solutions for a number of combinatorial optimization problems, such as spanning trees, minimum spanning trees, k-paths, shortest paths, k-matchings, etc. [15, 16, 19, 20, 21]. Most of the existing work focuses on finding diverse optimal solutions. In cases when finding the optimal solution is NP-complete, several works have focused on developing FPT algorithms [4, 16]. Nevertheless, as pointed out in [20], it would be more practical to consider finding a set of diverse "short" paths rather than one set of diverse shortest paths. They show that finding a set of approximately shortest paths with the maximum diversity is NP-hard, but leave the question of developing approximation algorithms open, a question that we answer in our paper for several problems. Similarly the problem of finding diverse maximum matchings was proved to be NP-hard in [15]. We remark that the main difference between our result and previous work is that our algorithms can find a diverse set of c-approximate solutions in polynomial time. If the attained diversity is not sufficient for the application, the user can input a larger c, in hopes of increasing it.

Multicriteria Optimization: In this domain, several optimization functions are given on a space of solutions. Clearly, there may not be a single solution that is the best for all objective functions, and researchers have focused on obtaining Pareto-optimal solutions, which are solutions that are non-dominated by other solutions. Put differently, a solution is Pareto-optimal if no other solution can have a better cost for all criteria. Since exact solutions are hard to find, research has focused on finding ϵ Pareto-optimal solutions, which are a $1 + \epsilon$ factor approximations of Pareto-optimal solutions. Papadimitriou and Yannakakis [35] showed that under pretty mild conditions, any mutlicriteria optimization problem admits an ϵ Pareto-optimal set of fully polynomial cardinality. In terms of being able to *find* such an ϵ Pareto-optimal set, they show that a (FPTAS) PTAS exists for the problem if and only if an associated GAP problem can be solved in (fully) polynomial time. Very recently, Herzel et al.[24] study the class of problems for which a FPTAS or PTAS exists for finding ϵ Pareto-optimal solutions that are *exact* in one of the criteria. Clearly such problems are a subset of the ones characterized by GAP. Herzel et al. [24] characterize the condition similarly: a FPTAS (PTAS) exists if and only if an associated DUALRESTRICT problem can be solved in (fully) polynomial time. For more details we refer the reader to the survey by Herzel at al. [25].

3   Problem Statement

First, we define some notations. We use the definition of optimization problems given in [2] with additional formalism as introduced in [18].

▶ **Definition 1** (Optimization Problem)**.** *An **optimization problem** Π is characterized by the following quadruple of objects $(I_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$, where:*

- I_Π *is the set of instances of Π. In particular for every $d \in \mathbb{N}$, $I_\Pi(d)$ is the set of instances of Π of input size at most d (bits);*

- Sol_Π *is a function that associates to any input instance $x \in I_\Pi$ the set of feasible solutions of x;*
- Δ_Π *is the measure function[3], defined for pairs (x, y) such that $x \in I_\Pi$ and $y \in \text{Sol}_\Pi(x)$. For every such pair (x, y), $\Delta_\Pi(x, y)$ provides a non-negative integer which is the value of the feasible solution y;*
- $\text{goal}_\Pi \in \{\min, \max\}$ *specifies whether Π is a maximization or minimization problem.*

We would like to identify a subset of our solution space which are (approximately) optimal with respect to our measure function. To this effect, we define a notion of approximately optimal feasible solution.

▶ **Definition 2** (Approximately Optimal Feasible Solution). *Let $\Pi(I_\Pi, \text{Sol}_\Pi, \Delta_\Pi, \text{goal}_\Pi)$ be an optimization problem and let $c \geq 1$. For every $x \in I_\Pi$ and $y \in \text{Sol}_\Pi(x)$ we say that y is a **c-approximate optimal solution of** x if for every $y' \in \text{Sol}_\Pi(x)$ we have $\Delta_\Pi(x, y) \cdot c \geq \Delta_\Pi(x, y')$ if $\text{goal}_\Pi = \max$ and $\Delta_\Pi(x, y) \leq \Delta_\Pi(x, y') \cdot c$ if $\text{goal}_\Pi = \min$.*

▶ **Definition 3** (Computational Problem). *Let $\Pi(I_\Pi, \text{Sol}_\Pi, \Delta_\Pi, \text{goal}_\Pi)$ be an optimization problem and let $\lambda : \mathbb{N} \to \mathbb{N}$. The **computational problem associated with** (Π, λ) is given as input an instance $x \in I_\Pi(d)$ (for some $d \in \mathbb{N}$) and real $c := \lambda(d) \geq 1$ find a c-approximate optimal feasible solution of x.*

▶ **Definition 4** (Diversity Computational Problem). *Let $\Pi(I_\Pi, \text{Sol}_\Pi, \Delta_\Pi, \text{goal}_\Pi)$ be an optimization problem and let $\lambda : \mathbb{N} \to \mathbb{N}$. Let $\sigma_{\Pi,t}$ be a diversity measure that maps every t feasible solutions of an instance of I_Π to a non-negative real number. The **diversity computational problem associated with** $(\Pi, \sigma_{\Pi,t}, k, \lambda)$ is given as input an instance $x \in I_\Pi(d)$ (for some $d \in \mathbb{N}$), an integer $k := k(d)$, and real $c := \lambda(d) \geq 1$, find k-many c-approximate solutions $y_1, \ldots, y_k$ to x which maximize the value of $\sigma_{\Pi,k}(x, y_1, \ldots, y_k)$.*

▶ **Proposition 5.** *Let $\Pi(I_\Pi, \text{Sol}_\Pi, \Delta_\Pi, \text{goal}_\Pi)$ be an optimization problem and let $\lambda : \mathbb{N} \to \mathbb{N}$. Let $\sigma_{\Pi,t}$ be a diversity measure that maps every t feasible solutions of an instance of I_Π to a non-negative real number. If the computational problem associated with (Π, λ) is $\mathcal{NP}$-hard then, the diversity computational problem associated with $(\Pi, \sigma_{\Pi,t}, \lambda)$ is also $\mathcal{NP}$-hard.*

Therefore the interesting questions arise when we compute problems associated with (Π, λ) which are in $\mathcal{P}$, or even more when, $(\Pi, \mathbb{1})$ is in $\mathcal{P}$ where $\mathbb{1}$ is the constant function which maps every element of the domain to 1. For the remainder of this paper, we will consider $\lambda(d)$ to be the constant function, and will simply refer to the constant as c.

Finally, we define bicriteria approximations for the diversity computational problem:

▶ **Definition 6** ((α, β) bi-approximation for the Diversity Computational Problem). *Consider the diversity computational problem associated with $(\Pi, \sigma_{\Pi,t}, k, c)$, and a given instance $x \in I_\Pi(d)$ (for some $d \in \mathbb{N}$). An algorithm is called an (α, β) bi-approximation for the diversity computational problem if it outputs k feasible solutions $y_1, \ldots, y_k$ such that a) y_i is a $\beta \cdot c$-approximate optimal feasible solution to x for all $1 \leq i \leq k$, and b) for any set $y_1', \ldots, y_k'$ of k-many c-approximate optimal feasible solutions, $\sigma_{\Pi,k}(y_1, \cdots, y_k) \cdot \alpha \geq \sigma_{\Pi,k}(y_1', \cdots, y_k')$. Furthermore, such an algorithm is said to run in polynomial time if the running time is polynomial in d and k.*

---

[3] We define the measure function only for feasible solutions of an instance. Indeed if an algorithm solving the optimization problem outputs a non-feasible solution then, the measure just evaluates to -1 in case of maximization problems and ∞ in case of minimization problems.

4 The Reduction: Enter Dispersion and Biobjective Optimization

As stated in the introduction, our problems are related to the classical dispersion problem in a metric space. Here we state the dispersion problem and show that under the exponential time hypothesis, 2-approximation is actually tight for the k-dispersion problem. We will then use dispersion to reduce the problem of finding diverse, approximately optimal solutions to solving an associated budget constrained optimization problems.

4.1 Dispersion Problem

▶ **Definition 7** (k-Dispersion, total distance). *Given a finite set of points P whose pairwise distances satisfy the triangle inequality and an integer $k \geq 2$, find a set $S \subseteq P$ of cardinality k so that $W(S)$ is maximized, where $W(S)$ is the sum of the pairwise distances between points in S.*

The main previous work on the k-dispersion problem relevant to us is [37], where the problem was named as Maximum-Average Facility Dispersion problem with triangle inequality (MAFD-TI). The problems are equivalent as maximizing the average distance between the points also maximizes the sum of pairwise distances between them and vice-versa.

The k-dispersion problem is $\mathcal{NP}$-hard, but one can find a set S whose $W(S)$ is at least a constant factor of the maximum possible in polynomial time by a greedy procedure [37]. We call the greedy procedure *furthest insertion*. It works as follows. Initially, let P be a singleton set that contains an arbitrary point from the given set. While $|S| < k$, add to S a point $x \notin S$ so that $W(S \cup \{x\}) \geq W(S \cup \{y\})$ for any $y \notin S$. Repeat the greedy addition until S has size k. The final S is a desired solution, which is shown to be a 4-approximation in [37]. It is worth noting that the furthest insertion in [37] initializes S as a furthest pair of points in the given set, and the above change does not worsen the approximation factor. In a later paper [7], the above greedy algorithm that chooses an arbitrary initial point is shown to be a 2-approximation, which is a tight bound for this algorithm [6].

▶ **Lemma 8** (Furthest Insertion in [7, 37]). *The k-dispersion problem can be 2-approximated by the furthest insertion algorithm.*

The running time of the furthest insertion algorithm is polynomial in $|S|$ the size of S, as it performs k iterations, each performing at most $O(k|S|)$ distance computations/lookups. Note that in our case S is the collection of objects of a certain type (matchings, paths, trees, etc.). Hence the size of our metric space is typically exponential in $|V|$ and $|E|$. This adds a new dimension of complexity to the traditional dispersion problems studied.

We prove that the k-dispersion problem cannot be approximated better than 2 given the Exponential Time Hypothesis.

▶ **Theorem 9** (Inapproximability of k-Dispersion). *Assuming the Exponential Time Hypothesis, for every $\varepsilon > 0$, no algorithm running in polynomial time can approximate the k-Dispersion problem to a factor of $2 - \varepsilon$.*

Proof. We use the inapproximability of the densest k subgraph problem.

▶ **Theorem 10** (Inapproximability of Densest k-subgraph with Perfect Completeness [30]). *Given the Exponential Time Hypothesis, for every $\delta > 0$, no algorithm running in polynomial time can distinguish between the following two cases, given as input a graph G and an integer k:*

Completeness: There is a clique of size k in G.

Soundness: *The subgraph induced by any k vertices in G has at most $\delta \cdot \binom{k}{2}$ many edges.*

Given a graph $G = (V, E)$, we define a distance function $\Delta : V \times V \to \mathbb{R}^{\geq 0}$ as follows.

$$\forall u, v \in V, \ \Delta(u, v) = \begin{cases} 0 \text{ if } u = v, \\ 2 \text{ if } (u, v) \in E, \\ 1 \text{ otherwise.} \end{cases}$$

Note that (V, Δ) is a metric space. Suppose there is a set of k vertices S such that it forms a clique in G, then, $\sum_{\{u,v\} \in \binom{S}{2}} \Delta(u, v) = 2 \cdot \binom{k}{2}$. On the other hand, if for some set S of k vertices we have that the subgraph induced by S in G has at most $\delta \cdot \binom{k}{2}$ many edges, then, $\sum_{\{u,v\} \in \binom{S}{2}} \Delta(u, v) \leq (1 + \delta) \cdot \binom{k}{2}$. The theorem statement thus follows from Theorem 10 by setting $\delta := \varepsilon / 2$. ◀

4.2 Reduction to Budget Constrained Optimization

Recall the definitions of the Diversity Computational Problem (Definition 4) and (a, b) bi-approximations (Definition 6). As the input instance $x \in I_\Pi$ will be clear from context, we drop the dependence on x, and assume a fixed input instance to a computational problem. Thus Sol_Π will denote the set of feasible solutions, and $\Delta_\Pi(y)$ the measure of the feasible solution y.

Diversity and similarity measures from metrics Let $d : \mathsf{Sol}_\Pi \times \mathsf{Sol}_\Pi \to \mathbb{R}^+$ be a metric on the space of feasible solutions. When such a metric is available, we will consider the diversity function $\sigma_{\Pi,t} : \mathsf{Sol}_\Pi \times \cdots \times \mathsf{Sol}_\Pi \to \mathbb{R}^+$ that assigns the diversity measure $\sum_{i,j} d(y_i, y_j)$ to a t-tuple of feasible solutions $(y_1, \cdots, y_t)$. Also, given such a metric d, define D to be the diameter of Sol_Π under d, i.e., $D = \max_{y,y' \in \mathsf{Sol}_\Pi} d(y, y')$. In many cases, we will be interested in the *similarity* measure $s_{\Pi,t}$ defined by $s_{\Pi,t}(y_1, \cdots, y_t) = \sum_{i,j} (D - d(y_i, y_j))$. The examples the reader should keep in mind are graph objects such as spanning trees, matchings, shortest paths, Hamiltonian circuits, etc., such that $d(y, y')$ denotes the Hamming distance, a.k.a. size of the symmetric difference of the edge sets of y and y', and s denotes the size of their intersection.

In the remainder of the paper we consider the above total distance (resp. similarity) diversity measures $\sigma_{\Pi,t}$ arising from the metric d (resp. similarity measure s), and we will parameterize the problem by d (resp. s) instead.

▶ **Definition 11** (Budget Constrained Optimization). *Given an instance of a computational problem Π, a $c \geq 1$, and a set $y_1, \ldots, y_i$ of feasible solutions in Sol_Π, define the metric budget constrained optimization problem $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ as follows:*

▪ *If $\mathsf{goal}_\pi = \min$, define $\Delta^* := \min_{y \in \mathsf{Sol}_\Pi} \Delta_\Pi(y)$. Then $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ is the problem*

$$\begin{aligned} \max \quad & f_d(y) := \sum_{j=1}^{i} d(y, y_i) \\ s.t. \quad & \Delta_\Pi(y) \leq c \cdot \Delta^* \\ & y \in \mathsf{Sol}_\Pi \setminus \{y_1, \ldots, y_i\} \end{aligned} \tag{1}$$

- *If* $\mathsf{goal}_\pi = \max$, *define* $\Delta^* := \max_{y \in \mathsf{Sol}_\Pi} \Delta_\Pi(y)$. *Then* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ *is the problem*

$$
\begin{aligned}
\max \quad & f_d(y) := \sum_{j=1}^{i} d(y, y_i) \\
s.t. \quad & \Delta_\Pi(y) \cdot c \geq \Delta^* \\
& y \in \mathsf{Sol}_\Pi \setminus \{y_1, \ldots, y_i\}
\end{aligned}
\tag{2}
$$

- *Define the similarity budget constrained problem* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$, *where* s *is a given similarity measure, with the same constraint set as above (depending on* goal_π*), but with the objective function changed to* $g_s(y) := \min \sum_{j=1}^{i} s(y, y_i)$ *instead of* $f_d(y) = \max \sum_{j=1}^{i} d(y, y_i)$.

▶ **Definition 12** (Bi-approximation to BCO). *An algorithm for an associated BCO is called an* (a, b) *bi-approximation algorithm if for any* $1 \leq i \leq k$, *it outputs a solution* y *such that the following holds.*

- *If* $\mathsf{goal}_\Pi = \min$ *and the associated BCO is* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$, *then a)* $y \in \mathsf{Sol}_\Pi \setminus \{y_1, \cdots, y_i\}$, *b)* $\Delta_\Pi(y) \leq b \cdot c \cdot \Delta^*$, *and c) for all* y' *satisfying the constraints of* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$, $f_d(y) \cdot a \geq f_d(y')$.
- *If* $\mathsf{goal}_\Pi = \max$ *and the associated BCO is* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$, *then a)* $y \in \mathsf{Sol}_\Pi \setminus \{y_1, \cdots, y_i\}$, *b)* $\Delta_\Pi(y) \cdot b \cdot c \geq \Delta^*$, *and c) for all* y' *satisfying the constraints of* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$, $f_d(y) \cdot a \geq f_d(y')$.
- *If* $\mathsf{goal}_\Pi = \min$ *and the associated BCO is* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$, *then a)* $y \in \mathsf{Sol}_\Pi \setminus \{y_1, \cdots, y_i\}$, *b)* $\Delta_\Pi(y) \leq b \cdot c \cdot \Delta^*$, *and c) for all* y' *satisfying the constraints of* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$, $g_s(y) \leq g_s(y') \cdot a$.
- *If* $\mathsf{goal}_\Pi = \max$ *and the associated BCO is* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$, *then a)* $y \in \mathsf{Sol}_\Pi \setminus \{y_1, \cdots, y_i\}$, *b)* $\Delta_\Pi(y) \cdot b \cdot c \geq \Delta^*$, *and c) for all* y' *satisfying the constraints of* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$, $g_s(y) \leq g_s(y') \cdot a$.

We are now ready to state our main theorem.

▶ **Theorem 13** (Reduction of DCP to BCO). *Consider an input* (Π, k, d, c) *to the diversity computational problem (DCP).*

- *For metric BCO,*

  1. *An* $(a, 1)$ *bi-approximation to* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ *can be used to give a* $(2a, c)$ *approximation to the DCP, and*
  2. *An* (a, b) *bi-approximation to* $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ *can be used to give a* $(4a, bc)$ *approximation to the DCP.*

- *For similarity BCO,*

  3. *A* $(1, 1)$ *bi-approximation to* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$ *can be used to give a* $(2, c)$ *approximation to the DCP,*
  4. *A* $(1, b)$ *bi-approximation to* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$ *can be used to give* $(4, bc)$ *approximation to the DCP,*
  5. *A* $(1 + \epsilon, 1)$ *bi-approximation to* $BCO(\Pi, (y_1, \ldots, y_i), c, s)$ *can be used to give* $(4, c)$ *approximation to the DCP, under the condition that the average pairwise distance in the optimal solution to the DCP is at least* $D \frac{4\epsilon}{1+2\epsilon}$.

In all of the above, the overhead for obtaining a bi-approximation for the DCP, given a bi-approximation for BCO problem, is $O(k)$.

Proof. Given a $BCO(\Pi, (y_1, \ldots, y_i), c, d)$, denote by $\Omega(c, (y_1, \ldots, y_i))$ its feasible set. When $b = 1$, the returned solution y lies in $\Omega(c, (y_1, \ldots, y_i))$. On the other hand when $b > 1$, the returned solution lies in the larger space $\Omega(bc, (y_1, \ldots, y_i))$. Our proof will handle these cases separately. We will first state two lemmas concerning the greedy heuristic of farthest point insertion, the first of which we suspect is folklore, but we could not find a reference. The second lemma is the heart of the argument, and is (as far as we know) novel.

▶ **Lemma 14.** *Let A be a metric space, and consider the k-dispersion problem in A as in Definition 7. Consider an oracle O that, given a set $P_i := \{p_1, \cdots, p_i\}$ of i points in A, returns a point $p \in A \setminus P_i$ such that $\sum_{j=1}^{i} d(p, p_i) \cdot a \geq \sum_{j=1}^{i} d(p', p_i)$ for all $p' \in A \setminus P_i$ for some constant $a > 1$. Then in k calls to O one can obtain a $2a$-approximate solution to the k-dispersion problem on A. In other words, the total pairwise distance between the k points of the solution obtained via O is at most that of the optimal solution to the k-dispersion divided by $2a$.*

Proof. We will prove the statement of the lemma for the average distance, which will imply the claim for the total distance. Let OPT be the average distance between solutions in the optimal solution to the k-dispersion problem. By Lemma 1 in [6], at any step i, there is always a solution whose distance to the already selected set P_i is at least $OPT/2$, and hence the same holds for the farthest solution. This implies that the point p returned by the algorithm has a distance at least $OPT/2a$. A simple inductive argument now finishes the proof: assume the lemma holds until stage i, i.e., the average pairwise distance between points in P_i is at least $OPT/2a$. When $p_{i+1} = p$ is chosen, its average distance to the i points in P_i is at least $OPT/2a$ by the above reasoning, and so the P_{i+1} has an average pairwise distance of at least $OPT/2a$. ◀

▶ **Lemma 15.** *Assume A is a metric space, and $B \subseteq A$. Suppose there is an oracle O that, given a set of i points $P_i := \{p_1, \cdots, p_i\}$ in A, outputs a point $p \in A \setminus P_i$ such that $\sum_{j=1}^{i} d(p_j, p) \geq \max_{p' \in B} \sum_{j=1}^{i} d(p_j, p')$. Then the oracle can be used to give a set of k points in A whose diversity is at least $1/4$ of an optimal solution to the k-dispersion problem in B.*

Proof. For a point $p \in A$, denote by $n_B(p)$ its closest point in B. By definition, if $p \in B$ then $n_B(p) = p$.

Assume that the oracle O in the hypothesis of the lemma exists. Then given the point set P_i, consider the point set $Q_i = \{n_B(p_j) : 1 \leq j \leq i\}$. As mentioned before, a careful analysis of Lemmas 3-6 in [6] reveals that there always exists a point $p^* \in B$ whose average distance to the points in Q_i is at least $OPT(B)/2$, where $OPT(B)$ denotes the optimal average pairwise distance between points in the optimal solution to the k dispersion problem on B.

Consider the three points $p_i, n_B(p_i)$ and p^*. By the triangle inequality, $d(p_i, p^*) \geq d(p^*, n_B(p_i)) - d(p_i, n_B(p_i))$. However, by definition, $d(p_i, n_B(p_i)) \leq d(p_i, p^*)$. This implies that $d(p_i, p^*) \geq d(p^*, n_B(p_i)) - d(p_i, p^*)$, implying that $d(p_i, p^*) \geq d(p^*, n_B(p_i))/2$. Summing over all i, we get that the total (and hence the average) distance of p^* to points P_i is at least half of its total distance to points in Q_i. Since the average distance of p^* to points in Q_i is at least $OPT(B)/2$, we get that the average distance of p^* to points in P_i is at least $OPT(B)/4$.

However, the oracle O returns a point $p \in A$ whose average distance to P_i is at least that of p^* to points in P_i, which as argued above is at least $OPT(B)/4$. By the same inductive

argument as in Lemma 14, we get that the average distance between points returned by k calls to O is at least $OPT(B)/4$, proving that it gives a 4-approximation. ◀

Given the above lemmas, we now prove the theorem.

- Type 1: An $(a, 1)$ bi-approximation to $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ returns a point in $\Omega(c, (y_1, \ldots, y_i))$. Therefore we only apply Lemma 14 and we are done.
- Type 2: An (a, b) bi-approximation to $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ returns a point in $\Omega(bc, (y_1, \ldots, y_i))$, that is an a-approximation to the farthest point in $\Omega(c, (y_1, \ldots, y_i))$. Applying Lemma 14 and Lemma 15 together, we get the claimed result.
- Type 3: A $(1, 1)$ bi-approximation to the similarity $BCO(\Pi, (y_1, \ldots, y_i), c, s)$ returns the farthest point in $\Omega(c, (y_1, \ldots, y_i))$, and so this case follows from the 2-approximation guarantee of the farthest insertion heuristic, without applying either lemma.
- Type 4: A $(1, b)$ bi-approximation to $BCO(\Pi, (y_1, \ldots, y_i), c, d)$ returns the farthest point in $\Omega(bc, (y_1, \ldots, y_i))$. In this case, we only apply Lemma 15 to get the claimed result.

We prove the remaining case of Type 5 separately. A $(1 + \epsilon, 1)$ bi-approximation to $BCO(\Pi, (y_1, \ldots, y_i), c, s)$ returns a point in $\Omega(c, (y_1, \ldots, y_i))$. However, *this point is not the farthest point, nor necessarily an approximation of it.* This is because the guarantee is only on the total similarity, and not the total distance. In other words, a $1 + \epsilon$ approximation to $g_s(y) := \sum_{j=1}^{i} s(y, y_i)$ is not a $1 + \epsilon$ approximation to $f_d(y) := \sum_{j=1}^{i} d(y, y_i)$– in fact, these functions are the "opposite" of each other, as $f_d(y) = Di - g_s(y)$.

Let y^* be the point in $\Omega(c, (y_1, \ldots, y_i))$ that maximizes $f_d(y)$, i.e., the farthest point. The $(1 + \epsilon, 1)$ bi-approximation algorithm returns a point y such that $g_s(y) \leq a \cdot g_s(y^*)$. Consider the condition that the average pairwise distance in the optimal solution OPT to the DCP is at least $D \frac{4\epsilon}{1+2\epsilon}$. We refer the reader now to Lemma 1 in [6], which proves that during farthest point insertion, at step $i + 1$ in the algorithm when $P_i = \{p_1, \cdots, p_i\}$ have been selected, there is a point $p \notin P_i$ whose average distance is at least $OPT/2$. While this lemma seems like it can only be applied to point sets P_i formed during an iterated farthest insertion, a careful analysis of Lemmas 3,4,5 and 6 in [6] used to prove it reveals that the argument does not demand that this be the case. That is, for any set P_i of i points, there always exist a $p \notin P_i$ whose average distance is at least $OPT/2$.

This implies the existence of a point $y' \in \Omega(c, (y_1, \ldots, y_i))$ such that the average distance of y' to the y_is is at least $OPT/2 \geq D \frac{2\epsilon}{1+2\epsilon} = D(1 - \frac{1}{1+2\epsilon})$. Since y^* is the farthest point, the same condition holds for y^*.

This implies that the total distance $f_d(y^*) \geq Di(1 - \frac{1}{1+2\epsilon})$, implying that $g_s(y^*) = Di - f_d(y^*) \leq Di/(1 + 2\epsilon)$. We then have the following string of inequalities

$$
\begin{aligned}
g_s(y^*) &\leq \frac{Di}{1 + 2\epsilon} \\
\iff Di &\geq (1 + 2\epsilon)g_s(y^*) \\
\iff \frac{Di}{2} &\geq (\frac{1}{2} + \epsilon)g_s(y^*) \\
\iff Di - (1 + \epsilon)g_s(y^*) &\geq \frac{1}{2}(D - g_s(y^*))
\end{aligned}
$$

But we know that the returned point y satisfies $g_s(y) \leq (1 + \epsilon)g_s(y^*)$, implying that $f_d(y) = Di - g_s(y) \geq \frac{1}{2}(D - g_s(y^*)) = f_d(y^*)/2$. Hence y is a 2-approximation to the farthest point, and we can apply Lemma 14 to complete the proof. ◀

A few remarks are in order:

- The above theorem provides a recipe for solving the diversity computational problem for any given optimization problem. As long as *either* the metric or the similarity budget constrained optimization problems can be solved or approximated in polynomial time, one has an analogous result for the DCP.
- In the remainder of this paper we will see several application that follow from the above 5 "types" of bi-approximations available. These include DCP for Maximum Matching (Type 1), DCP for shortest path (Type 3), DCP for minimum weight bases of a matroid, minimum spanning trees (Types 4 and 5).
- Whenever either a or b (or both) is set to be $1 + \epsilon$, we call a bi-approximation for the BCO problem an FPTAS if the running time is polynomial in $1/\epsilon$ in addition to being polynomial in d and i. Otherwise we call it a PTAS.

Relation to Multicriteria Optimization: Observe that for similarity BCOs, we need either a or b to be 1. This class of biobjective problems that have a PTAS that is exact in one of the criteria is a special case of the multicriteria problems that have a PTAS that is exact in one of the criteria. Herzel et al. [24] showed that this class is exactly the class of problems for which the DUALRESTRICT version of the problem, posed by Diakonikolas and Yannakakis [12]), can be solved in polynomial time. These are also the class of problems having a polynomial-time computable approximate ϵ-Pareto set that is exact in one objective. This equivalence means that our theorem is applicable to this entire class of problems.

4.3   Relaxed BCOs and Self-Avoidance

Before we delve into our applications, we describe another challenge in directly applying results from multicriteria optimization literature. For a BCO, the second constraint demands that $y \in \mathsf{Sol}_\Pi \setminus \{y_1, \cdots, y_i\}$. Intuitively y is the farthest point to the set of already discovered solutions $\{y_1, \cdots, y_i\}$, and because it is defined implicitly, without the second constraint y may equal one of the y_j ($1 \le j \le i$). Consider an alternate BCO, which we call BCO^r where the constraint is relaxed to $y \in \mathsf{Sol}_\Pi$. For many graph problems, solving BCO^r combined with the approach by Lawler [28] gives a solution to the original BCO. This is extremely useful because most of the literature on multicriteria optimization concerns optimization of the relaxed type of problems BCO^r, and one can borrow results derived before without worrying about the second constraint. We remark that for other problems, k-best enumeration algorithms (see [17, 22, 28, 29, 33] for examples) may be useful to switch from the BCO to its relaxed version. *Thus any algorithm for BCO^r can be used, modulo the self-avoiding constraint (to be handled using Lawler's approach), to give a polynomial time algorithm for the Diversity Computational Problem with the same guarantees as in Theorem 13.* We provide examples of the approach by Lawler in subsequent sections where we consider specific problems.

5   Application 1: Diverse Spanning Trees

In this section, we discuss the diverse spanning trees problem, which is the diversity computational problem for spanning trees with Hamming distance function as the diversity measure. Let $G = (V, E)$ be an undirected graph. The problem aims to output a set S of k spanning trees $T_1, \cdots, T_k$ of G such that the sum of the pairwise distances $\sum_{i,j \in S} d(T_i, T_j)$ is maximized, where d is the Hamming distance between the edge sets of the trees. While this problem actually has an exact algorithm running in time $O((kn)^{2.5}m)$ [21], we get a faster approximation algorithm.

▶ **Theorem 16.** *Given a simple graph $G = (V, E)$, there exists an $O(knm\alpha(n,m))$-time algorithm, where $\alpha(\cdot)$ is the inverse of the Ackermann function, that generates k spanning trees $T_1, \cdots, T_k$, such that the sum of all pairwise Hamming distances is at least half of an optimal set of k diverse spanning trees.*

Proof. We develop an exact $(1,1)$ polynomial time subroutine for the associated BCO problem. Using our reduction Theorem 13, we observe that since one is only required to output spanning trees, the associated budget constrained problem has no inequalities. Having obtained i trees $T_1, \cdots, T_i$ (starting with an arbitrary spanning tree T_1), the BCO problem looks like

$$
\begin{aligned}
\max \quad & f_d(y) := \sum_{j=1}^{i} d(T, T_i) \\
\text{s.t.} \quad & T \in \mathsf{Sol}_\Pi \setminus \{T_1, \ldots, T_i\}
\end{aligned}
\tag{3}
$$

The relaxed budget constrained problem then simply asks to maximize $\sum_{j=1}^{i} d(T, T_i)$. We first show how to solve this problem exactly, and then adapt the approach by Lawler [28] to handle the self-avoiding constraint. The algorithm to maximize $\sum_{j=1}^{i} d(T, T_i)$ is very simple: give each edge e a weight $w(e) = \sum_{j=1}^{i} \mathbb{1}(e \in T_j)$ and compute the minimum spanning tree T with respect to these edge weights.

▶ **Lemma 17.** *The minimum spanning tree T with respect to the edge weights w satisfies $\sum_{j=1}^{i} d(T_j, T) \geq \sum_{j=1}^{i} d(T_j, T')$ for any spanning tree T'.*

Proof. Given input spanning trees $T_1, \cdots, T_i$, suppose T^A is the tree returned by the algorithm in Section 4.1, and T^* is an optimal spanning tree. Let E_j be the subset of edges with weight j. That is, $E_j = \{e \in E | w(e) = j\}$ for all $0 \leq j \leq i$ and partition E as $E = E_0 \cup E_1 \cup \cdots \cup E_i$.

Let $c_j = |T^A \cap E_j|$ and $\beta_j = |T^* \cap E_j|$. To prove that T^A is optimal, we will show that $\sum_{j=1}^{i} d(T^A, T_j) \geq \sum_{j=1}^{i} d(T^*, T_j)$.

Define $A = \sum_{j=1}^{i} d(T^A, T_j)$ and $B = \sum_{j=1}^{i} d(T^*, T_j)$. Plugging α_j and β_j into the sum of pairwise distances from our tree and the optimal tree to constructed trees, we get

$$
A = \sum_{j=1}^{i} d(T^A, T_j) = (n-1)i - \sum_{j=1}^{i} j\alpha_j
$$

and

$$
B = \sum_{j=1}^{i} d(T^*, T_j) = (n-1)i - \sum_{j=1}^{i} j\beta_j.
$$

We will show that $A - B = \sum_{j=0}^{i} j(\beta_j - \alpha_j) \geq 0$. We first express $A - B$ as the dot product of two vectors

$$
A - B = <0, 1, \cdots, i> <\beta_0 - \alpha_0, \beta_1 - \alpha_1, \cdots, \beta_i - \alpha_i>^t,
$$

where $<.>^t$ denotes transpose, and makes a column vector out of a row vector. We develop some more notation:

$$
E^j := E_0 \cup E_1 \cup \cdots \cup E_j
$$

$$\alpha^j := \alpha_0 + \alpha_1 + \cdots + \alpha_j$$

$$\beta^j := \beta_0 + \beta_1 + \cdots + \beta_j$$

$$c^j := <0, 1, \cdots, j>$$

$$\alpha_v^j := <\alpha_0, \alpha_1, \cdots, \alpha_j>^t$$

$$\beta_v^j := <\beta_0, \beta_1, \cdots, \beta_j>^t$$

With this we have that $A - B = c^i(\beta_v^i - \alpha_v^i)$.

We will show $c^i\beta_v^i \geq c^i\alpha_v^i$ by induction on i, which will prove that $A \geq B$, completing the proof.

We first claim that $\alpha^j \geq \beta^j$ for all $0 \leq j \leq i$. To prove this, fix a j and let C_1 be the number of connected components (including any isolated vertices) of $T^A \cap E^j$ and C_2 be the number of connected components of $T^* \cap E^j$. Clearly, $C_1 \leq C_2$ since for Kruskal's algorithm for minimum spanning trees is the greedy algorithm that adds as many edges from E^j into the solution as possible without creating a cycle, and the addition of an edge decreases the number of connected components by one. Since, $C_1 = n - \alpha^j$ and $C_2 = n - \beta^j$, $C_1 \leq C_2$ implies that $\alpha^j \geq \beta^j$.

Now, we continue our proof that $c^i\beta_v^i \geq c^i\alpha_v^i$ by induction on i. Observe $||\alpha_v^i||_1 = ||\beta_v^i||_1 = n' := n - 1$, as both spanning trees contain exactly $n - 1$ edges.

Base Case For the base case of $i = 1$, $c^1 = <0, 1>$, $\alpha_v^1 = <\alpha_0, n' - \alpha_0>$, and $\beta_v^1 = <\beta_0, n' - \beta_0>$. From the previously mentioned property, we know $\alpha_0 = \alpha^0 \geq \beta^0 = \beta_0$. Hence, $c^1\beta_v^1 = n' - \beta_0 \geq n - \alpha_0 = c^1\alpha_v^1$.

The induction hypothesis assumes that $c^i\beta_v^i \geq c^i\alpha_v^i$ for $i = K - 1$, i.e., the statement is true for two vectors of length $K - 1$. Now, consider when $i = K$. Since $||\alpha_v^{K-1}||_1 = \alpha^{K-1} \geq ||\beta_v^{K-1}||_1 = \beta^{K-1}$, there are two cases,

Case 1): $||\beta_v^{K-1}||_1 = ||\alpha_v^{K-1}||_1$ Then since $||\beta_v^K||_1 = ||\alpha_v^K||_1$, we get that $\alpha_K = \beta_K$. Using the induction hypothesis, we get

$$c^K\beta_v^K = c^{K-1}\beta_v^{K-1} + K\beta_K \geq c^{K-1}\alpha_v^{K-1} + K\alpha_K = c^K\alpha_v^K.$$

Case 2) $||\alpha_v^{K-1}||_1 > ||\beta_v^{K-1}||_1$. Then $||\alpha_v^K||_1 = ||\beta_v^K||_1$ yields $\alpha_K < \beta_K$. Let $d = \beta_K - \alpha_K = ||\alpha_v^{K-1}||_1 - ||\beta_v^{K-1}||_1$. Then, $c^K\beta_v^K - c^K\alpha_v^K = c^{K-1}(\beta_v^{K-1} - \alpha_v^{K-1}) + Kd$. We claim that this quantity is positive. This is because the minimum value of $c^{K-1}(\beta_v^{K-1} - \alpha_v^{K-1})$, subject to the conditions that a) $||\alpha_v^{K-1}||_1 - ||\beta_v^{K-1}||_1 = d > 0$ and b) $||\alpha_v^K||_1 = ||\beta_v^K||_1$, occurs when $\beta_j = \alpha_j$ for all $0 \leq j \leq K - 2$, $\beta_{K-1} = \alpha_{K-1} - d$, and $\beta_K = \alpha_K + d$. For this setting, the value of $c^{K-1}(\beta_v^{K-1} - \alpha_v^{K-1}) + Kd$ equals $-(K-1)d + Kd = d$, which is positive, completing the proof.   ◀

If $T \neq T_j$ for all $1 \leq j \leq i$, that is, the new spanning tree is different from all previous trees, then we are done and Theorem 16 is proved using Lemma 17.

Self-Avoiding Constraint However, this is not guaranteed as our measure is the sum-of-distances and not the minimum distance. Note that for furthest point insertion, we need the point that is furthest from the current set, but does not belong to the current set. This is an issue we face because of the implicit nature of the furthest point procedure in the exponential-sized space of spanning trees: in the metric k-dispersion problem, it was easy to guarantee distinctness as one only considered the $n - i$ points not yet selected.

We now show how to guarantee that the new tree is distinct. In case that $T = T_j$ for some $1 \leq j \leq i$, we use the approach by Lawler [28]. We will obtain $i + 1$ distinct spanning trees $\widetilde{T}_1, \cdots, \widetilde{T}_{i+1}$, at least one of which must then be distinct from $T_1, \cdots, T_i$, which will

be chosen as T_{i+1}. Set $\widetilde{T}_1 = T_j$. For every edge $e \in \widetilde{T}_1$, we find the minimum spanning tree T_e (with respect to the same edge weights $w(e)$ as before) after deleting e from the graph. Among the collection $\{T_e\}_{e \in \widetilde{T}_1}$ of trees thus obtained, we find the one whose sum of distances from $T_1, \cdots, T_i$ is as large as possible, and set it to be $\widetilde{T}_2$. Note that $\widetilde{T}_2 \neq \widetilde{T}_1$. If $\widetilde{T}_2 \neq T_j$ for all $1 \leq j \leq i$, then we are done; else, we repeat the above procedure to obtain the collection $\{T_e\}_{e \in \widetilde{T}_2}$ and set $\widetilde{T}_3$ as the best one in $\{T_e\}_{e \in \widetilde{T}_1} \cup \{T_e\}_{e \in \widetilde{T}_2} \setminus \{\widetilde{T}_1, \widetilde{T}_2\}$, after which we are either done or we repeat to get $\widetilde{T}_4$, and so on. We stop after obtaining $\widetilde{T}_{i+1}$, and select the $\widetilde{T}_{(.)}$ that is distinct from all of $T_1, \cdots, T_i$. The proof of Lemma 17 implies that the T_{i+1} thus obtained is the furthest tree from the $T_1, \cdots, T_i$ among all trees $T \notin \{T_1, \cdots, T_i\}$, and we have accomplished furthest point insertion in the space of spanning trees. In total, the above needs to compute $O(kn)$ instances of the minimum spanning tree, so the running time is $O(knm\alpha(n, m))$ [9]. ◀

6 Application 2: Diverse Approximate Shortest Paths

Given a graph $G = (V, E)$, non-negative edge weights $w(e)$, two vertices s and t, and a factor $c > 1$, the diversity computational problem asks to output k many st paths, such that the weight of each path is within a factor c of the weight of the shortest st path, and subject to this constraint, the total pairwise distance between the paths is maximized. Here the distance between two paths is again the Hamming distance, or size of symmetric difference of their edge sets.

In [20], it is shown that finding k shortest paths with the maximum diversity (i.e. the average Hamming distance between solutions) can be solved in polynomial time, but finding k "short" paths with the maximum diversity is NP-hard. In contrast, in what follows, we will show that finding k "short" paths with constant approximate diversity is polynomial-time solvable.

We will show that the associated budget constrained optimization problem for this is of Type 3 in Theorem 13. In other words, we will show that the BCO can be solved exactly. This will result in a $(2, c)$ approximation algorithm for the diversity computational problem.

Hence, we need an algorithm that implements: given a set S of c-st-shortest paths, find a c-st-shortest path $P \notin S$ so that $W(S \cup \{P\})$ is maximum among all $W(S \cup \{P'\})$ for c-st-shortest path $P' \notin S$. Here, $W(S')$ is the sum of all pairwise Hamming distances between two elements in S'. This is a special case of the bicriteria shortest paths, for which there is an algorithm in [34]. In our case, **one of the two weight functions is an integral function with range bounded in** $[0, k]$. Hence, it can be solved more efficiently than the solution in [34], which can be summarized as following.

▶ **Lemma 18** (Exact solution to the relaxed BCO^r problem). *Given a real $c \geq 1$ and a directed simple graph $G = (V \cup \{s, t\}, E)$ associated with two weight functions on edges $\omega : E \to \mathbb{R}^+$ and $f : E \to \{0, 1, \ldots, r\}$, there is an $O(r|V|^3)$-time algorithm to output an st-path P^* so that $\sum_{e \in E(P^*)} f(e)$ is minimized while retaining $\sum_{e \in E(P^*)} \omega(e) \leq c \sum_{e \in E(P)} \omega(e)$ for all st-paths P.*

Proof. Let $D(.,.)$ be a $|V| \times r(|V| - 1)$ table with the entry values defined as the following:

$$D(v, c) = \min \left\{ \sum_{e \in E(P)} \omega(e) : P \text{ is an } sv\text{-path with } \sum_{e \in E(P)} f(e) \leq c \right\}.$$

If the RHS takes the minimum from an empty set, then we define $D(v, c) = \infty$. By definition, P^* has $\sum_{e \in E(P^*)} f(e) = c^*$ so that

$$c^* = \min\{c \in [0, r(|V| - 1)] : D(t, c) \leq c \cdot dis_\omega(s, t)\}$$

where $dis_\omega(s, t)$ denotes the distance under ω from s to t. By backtracking from the $D(t, c^*)$, one can construct P^* from the table D in $O(r|V|)$ time. Hence, our goal is to fill out the entries in D using $O(r|V|^3)$ time.

To compute $D(v, 0)$ for all $v \in V$, it suffices to run Dijkstra's algorithm [13] on the graph obtained from G with the removal of the edges whose $f(e) > 0$. This step takes $O(|V|^2)$ time.

Given $D(v, c')$ for all $v \in V, c' \in [0, c-1]$, to compute $D(v, c)$ for all $v \in V$ there are two cases to discuss. We say an sv-path P *admits* $D(v, c')$ if P has $\sum_{e \in E(P)} \omega(e) = D(v, c')$ and $\sum_{e \in E(P)} f(e) \leq c'$.

Case 1. Some sv-path $(s, \ldots, x, v)$ with $f((x, v)) > 0$ admits $D(v, c)$.
Case 2. Some sw-path $P = (s, \ldots, y, w)$ with $f((y, w)) > 0$ admits $D(w, c)$, and P together with some edges e whose $f(e) = 0$ admits $D(v, c)$.

For Case 1, it suffices to check $D(x, c - f((x, v)))$ for all the neighbors of v with $f((x, v)) > 0$. This step takes $O(|V|)$ time. For Case 2, observe that $D(w, c)$ can be determined by Case 1. It suffices to initialize $D(v, c)$ for all $v \in V$ as the values obtained from Case 1 and update $D(v, c)$ if there is a path Q from some node w to v so that every edge $e \in E(Q)$ has $f(e) = 0$ and $D(w, c) + \sum_{e \in E(Q)} \omega(e) < D(v, c)$. This can be implemented in $O(|V|^2)$ time for each c by running Dijkstra's algorithm on the graph $G = (V, E \cup E_1 \setminus E_2)$ where E_1 is a set of new directed edges (s, v) for all $v \in V$ with $f((s, v)) = 0$ and $\omega((s, v))$ as the initial $D(v, c)$ obtained from Case 1, and E_2 is the set of all the edges $e \in E$ that have $f(e) > 0$. It is worth noting that, though we do not explicitly check whether the found paths contain cycles or not (i.e. not simple paths), the paths found to admit $D(v, c)$ must be simple since all shortest paths in a positive weight graph are simple.

To sum up, an optimal solution P^* can be found in $O(r|V|^3)$ time.   ◄

Self-Avoiding Constraint We now turn to solving the associated (non-relaxed) *BCO* problem, by generalizing the above lemma to Corollary 19. Thus Corollary 19 will help us avoid the situation that a furthest insertion returns a path that is already picked by some previous furthest insertion.

▶ **Corollary 19** (Exact solution to the *BCO* problem). *Given a real $c \geq 1$, a directed simple graph $G = (V \cup \{s, t\}, E)$ associated with two weight functions on edges $\omega : E \to \mathbb{R}^+$, $f : E \to \{0, 1, \ldots, r\}$, and two disjoint subsets of edges $E_{in}, E_{ex} \subseteq E$ so that all edges in E_{in} together form a directed simple path P_{prefix} starting from node s, there exists an $O(r|V|^3)$-time algorithm to output an c-st-shortest path P^* under ω so that $\sum_{e \in E(P^*)} f(e)$ is minimum among all the c-st-shortest paths P that contain P_{prefix} as a prefix and contain no edges from E_{ex}, if such an c-st-shortest path exists.*

Proof. Let $P_{\text{prefix}} = (s, \ldots, v)$. This suffices to find a feasible vt-path $P_{\text{suffix}} = (v, \ldots, t)$ on the graph obtained from G with the removal of all the nodes in P_{prefix} except v and with the removal of all edges in E_{ex} and those incident to the deleted nodes. This can be computed by Lemma 18.   ◄

We are ready to state our main result for the diverse c-st-shortest paths.

▶ **Theorem 20** ($(2,c)$ bi-approximation to the Diversity Problem on Shortest Paths). *For any directed simple graph $G = (V \cup \{s,t\}, E)$, given a $c > 1$ and a $k \in \mathbb{N}$, there exists an $O(k^3|V|^4)$-time algorithm that, if G contains at least k distinct c-st-shortest paths, computes a set S of k distinct c-st-shortest paths so that the sum of all pairwise Hamming distances between two paths in S is at least one half of the maximum possible; otherwise, reports "Non-existent."*

Proof. Let P_1 be an arbitrary 1-st-shortest path (i.e. a shortest path from s to t), which can be computed in $O(|V|^2)$ time by Dijkstra's algorithm. We perform the furthest insertion mentioned in Lemma 8 to obtain the c-st-shortest paths $P_2, P_3, \ldots, P_k$. The collection $\{P_i : i \in [1,k]\}$ is a desired solution.

To perform the i-th furthest insertion for $i \in [1, k-1]$, we need an algorithm for the problem that, given $\{P_j : 1 \le j \le i\}$, find an c-st-shortest path P_{i+1} so that the following conditions both hold:

1. $\sum_{e \in E(P_{i+1})} f_i(e)$ is minimum among all c-st-shortest paths where $f_i(e)$ for $e \in E$ is defined as the number of times that e appears in P_j for all $j \in [1,i]$. Hence, f_i maps E to $\{0, 1, \ldots, i\}$.
2. $P_{i+1} \ne P_j$ for all $j \in [1,i]$.

To find an c-st-shortest path that satisfies the first condition, it is an instance of the bicriteria shortest path problem with two weight functions ω and f_i. By Lemma 18, this can be done in $O(i|V|^3)$ time.

However, the c-st-shortest path found by minimizing $\sum_e f_i(e)$ is not necessarily different from those found in the previous furthest insertions. To remedy, one can find the c-st-shortest paths whose $\sum_e f_i(e)$ are the j-th smallest for all $j \in [1, i+1]$. Some of the $i+1$ c-st-shortest paths can meet the second condition. This step can be implemented by the standard approach due to Lawler [28]. That is, suppose in time T one can compute P_{i+1} with an additional constraint that some given edge set $E_{i,in}$ (all edges in $E_{i,in}$ together form a directed path starting from node s) must be contained in $E(P_{i+1})$ and some given edge set $E_{i,ex}$ must not be contained in $E(P_{i+1})$, then enumerating the $i+1$ smallest candidates for P_{i+1} can be done in $O(i|V|T)$ time. By Corollary 19, $T = O(i|V|^3)$. Thus, the i-th furthest insertion can be done in $O(i^2|V|^4)$ time. In total, our approach needs $O(k^3|V|^4)$ time as desired. ◀

7 Application 3: Diverse Approximate Maximum Matchings

Consider the diversity computational problem for computing k many c maximum matchings for undirected graphs. In [15], the authors present an algorithm, among others, to find a pair of maximum matchings for bipartite graphs whose Hamming distance is maximized. In contrast, our result can be used to find $k \ge 2$ approximate maximum matchings for any graph whose diversity (i.e. the average Hamming distance) approximates the largest possible by a factor of 2.

Since $\mathsf{goal}_\Pi = \max$, using the Hamming metric on the edge set of the solutions gives us a BCO with minimization in the objective function (the sum of the total distances) and constraints with lower bound.

We show that this problem can be restated into the budgeted matching problem [5]. As noted in [5], though the budgeted matching is in general $\mathcal{NP}$-hard, if both the weight and cost functions are integral and have a range bounded by a polynomial in $|V|$, then it can be solved in polynomial time with a good probability by a reduction to the exact perfect matching problem [8, 32]. The exact running time for such a case is not stated explicitly

in [5]. We combine the algorithm in [5] and the approach by Lawler [28] to prove our main theorem for diverse matchings (Theorem 22).

Define the distance between two matchings as the Hamming distance between the edge sets. The associated relaxed BCO^r is a maximizatin problem with lower bounded constraints. It can be restated as: given a non-empty set S of c-maximum matchings, find a c-maximum matching $M \notin S$ so that $W(S \cup \{M\})$ is maximum among all $W(S \cup \{M'\})$ for c-maximum matching $M' \notin S$ (here $W(X)$ is the sum of pairwise distances between all matchings in X. This problem can be restated into the budgeted matching problem [5]. As noted in [5], though the budgeted matching is in general $\mathcal{NP}$-hard, if both the weight and cost functions are integral and have a range bounded by a polynomial in $|V|$, then it can be solved in polynomial time with a good probability by a reduction to the exact perfect matching problem [8, 32]. The exact running time for such a case is not stated explicitly in [5], and we analyze it as below.

▶ **Lemma 21** (Restricted Budgeted Matching Problem). *Given an undirected simple graph $G = (V, E)$ and a cost function $c : E \to \{0, 1, \ldots, r\}$ on edges, there exists an $O(r^2 |V|^6 \log^2 r|V|)$-time randomized algorithm that can find a matching of smallest cost among all c-maximum matchings with some constant success probability where the cost of a matching is the sum of the costs of all edges in the matching.*

Proof. Suppose that, for all $x \in [0, X]$ with $X = O(|V|)$, for all $y \in [0, Y]$ with $Y = O(r|V|)$, we can decide whether G contains a matching M that consists of x edges and has cost y. Then, the desired matching M^* can be found.

As the reduction in [5], we construct an edge-weighted graph $H = (U, F)$ so that

- $U = V \cup \{z_1, z_2, \ldots, z_{|V|}\}$ where $z_i \notin V$ for all $i \in [1, |V|]$,
- $F = E \cup \{\{z_i, z_j\} : i \neq j \in [1, |V|]\} \cup \{\{x, z_i\} : x \in V, i \in [1, |V|]\}$, and
- for every edge $e \in F$, if e is also in E, then it has weight $\Gamma + c(e)$ for some sufficiently large integer Γ to be determined later; otherwise, it has weight 0.

By setting $\Gamma \geq r\lceil |V|/2\rceil + 1$, G has a matching M that consists of x edges and has cost y if and only if H has a perfect matching of weight $x\Gamma + y$. This is an instance of the exact perfect matching problem [8, 32], which can be solved by a randomized algorithm in $O(r|V|^4 \log r|V|)$ time with some constant success probability. Summing over all choices of x and y, the running time is $O(r^2 |V|^6 \log^2 r|V|)$. ◀

We are ready to prove our main result for the diverse c-maximum matchings.

▶ **Theorem 22.** *There exists a $O(k^4 |V|^7 \log^3 k|V|)$ time, $(2, c)$ bi-approximation to the diversity computational problem for c-maximum matchings, with failure probability $1/|V|^{\Omega(1)}$.*

Proof. Let M_1 be an arbitrary maximum matching, which can be computed in $O(|V|^{2.5})$ time [31]. For each $i \in [1, k-1]$, given $S_i = \{M_j : j \in [1, i]\}$, find a c-maximum matching $M_{i+1} \notin S_i$ so that $W(S_i \cup M_{i+1})$ is maximum among all $W(S_i \cup M')$ for c-maximum matching $M' \notin S_i$. By Lemma 21, set the cost function $c_i(e)$ for every $e \in E$ as the number of times that edge e appears in M_j for all $j \in [1, i]$ and apply the algorithm for Lemma 21 to solve the instance $(G = (V, E), c_i, c)$. This returns an c-maximum matching $M^\dagger$ so that $W(S_i \cup M^\dagger)$ is maximum among all $W(S_i \cup M')$ that M' is an c-maximum matching and may or may not be contained in S_i. Thus we need to guarantee self-avoidance.

To remedy, we enumerate the best $i+1$ candidates for M_{i+1}. Some of them is not contained in S_i. This enumeration can be done by running the algorithm for Lemma 21 $O(i|V|)$ times,

each of which preselects some edges to be included in and some to be excluded from M_{i+1}, as the Lawler's approach [28]. For each $i \in [1, k-1]$, this step takes $O(k^3 |V|^7 \log^3 k|V|)$ time and may fail with probability $(k|V|)^{-\Omega(1)}$. The failure probability is $(k|V|)^{-\Omega(1)}$ rather than a constant because we can run the algorithm that may err $O(\log k|V|)$ times.

Summing up the running time for the $k-1$ furthest insertions, the total running time is $O(k^4 |V|^7 \log^3 k|V|)$, and the failure probability is $1/|V|^{\Omega(1)}$ by the Union bound. ◄

8 Application 4: Diverse Minimum Weight Matroid Bases and Minimum Spanning Trees

One of the original ways to attack the peripatetic salesman problem (Krarup [26]) was to study the k edge-disjoint spanning trees problem [10]. Note that the existence of such trees is not guaranteed, and one can use our results in Section 5 to maximize diversity of the k trees found.

However, for an application to the TSP problem, cost conditions must be taken into account. Here we study the diverse computational problem (DCP) on minimum spanning trees: Given a weighted undirected graph $G = (V, E)$ with nonnegative weights $w(e)$, $c > 1$ and a $k \in \mathbb{N}$, return k spanning trees of G such that each spanning tree is a c-approximate minimum spanning tree, and subject to this, the diversity of the k trees is maximized. Here again the diversity of a set of trees is the sum of pairwise distances between them, and the distance between two trees is the size of their symmetric difference.

Our results in this section generalize to the problem of finding k diverse bases of a matroid such that every basis in the solution set is a c approximate minimum-weight basis. The DCP on MSTs is a special case of this problem. However, in order to not introduce extra notation and definitions here, we will describe our method for minimum spanning trees. We will then briefly sketch how to extend the algorithm to the general matroid case.

Starting with $T_1 = MST(G)$ (a minimum spanning tree on G, computable in polynomial time), assume we have obtained i trees $T_1, \cdots, T_i$, all of which are c-approximate minimum spanning trees. Assign to each edge a length $\ell(e)$ which equals $|j : 1 \le j \le i, e \in T_j|$.

▷ **Claim 23.** Given $T_1, \cdots, T_i$, finding T_{i+1} that maximizes $\sum_{j=1}^{i} d(T, T_j)$ is equivalent to finding T that minimizes $\sum_{e \in T} \ell(e)$.

Proof. An explicit calculation reveals that $\sum_{e \in T} \ell(e) = (n-1)i - \sum_{j=1}^{i} d(T, T_j)$. ◄

Consider now the associated *similarity budget constrained optimization problem*

$$
\begin{aligned}
\min \quad & \sum_{e \in T} \ell(e) \\
\text{s.t.} \quad & w(T) \le c \cdot w(MST(G)) \\
& T \in \mathsf{Sol}_\Pi \setminus \{T_1, \ldots, T_i\}
\end{aligned}
\tag{4}
$$

Here Sol_Π is just the set of spanning trees on G. We will handle the self-avoiding constraints in a similar fashion as in Section 5. For the moment consider the relaxed BCO^r where the last constraint is simply $T \in \mathsf{Sol}_\Pi$. This is a budget constrained MST with two weights. This problem has been considered by Ravi and Goemans [36], who termed it the CMST problem. They provide a $(1, 2)$ bi-approximation that runs in near-linear time, and a $(1, 1 + \epsilon)$ bi-approximation that runs in polynomial time[4]. Also, they show that the $(1, 1 + \epsilon)$

---

[4] The latter is a PTAS, not an FPTAS.

bi-approximation can be used as a subroutine to compute a $(1 + \epsilon, 1)$ bi-approximation in pseudopolynomial time.

Applying their results and observing that we are in cases 4 and 5 of Theorem 13, we get

▶ **Theorem 24** (DCP for Mininum Spanning Trees). *There exists a*

- *polynomial (in n, m and k) time algorithm that outputs a $(4, 2c)$ bi-approximation to the DCP problem for MSTs.*
- *polynomial (in n, m and k) and exponential in $1/\epsilon$ time algorithm that outputs a $(4, (1+\epsilon)c)$ bi-approximation to the DCP problem for MSTs.*
- *pseudopolynomial time algorithm that outputs a $(4, c)$ bi-approximation to the DCP problem for MSTs, as long as the average distance between the trees in the optimal solution to the k DCP on c-approximate minimum spanning trees does not exceed $\frac{4\epsilon(n-1)}{1+2\epsilon}$.*

Extension to Matroids: It is stated in the paper by Ravi and Goemans [36] that the same result holds if one replaces the set of spanning trees by the bases of any matroid. It is straightforward to show that the analog of Claim 23 hold in the matroid setting too. With a bit of work, one can also generalize the approach of Lawler [28] to avoid self-intersection (the bases found so far), and thus all the techniques generalize to the matroid setting. In all of this, we assume an independence oracle for the matroid, as is standard. In [16], it is shown that, given integers k, d, finding k *perfect* matchings so that every pair of the found matchings have Hamming distance at least d is NP-hard. This hardness result also applies to finding weighted diverse bases and weighted diverse common independent sets.

9   Conclusion

We obtained bi-approximation algorithms for the diversity computational problems associated with approximate minimum-weight bases of a matroid, shortest paths, matchings, and spanning trees. Our general reduction lemma can be applied to any computational problem for which the DUALRESTRICT version defined by Papadimitriou and Yannakakis [35] can be solved in polynomial time. There are (at least) three open questions:

- Are there other interesting examples of problems to which our reduction lemma applies?
- It is noteworthy that while Types 3, 4 and 5 in Theorem 13 contain the class for which DUALRESTRICT can be solved efficiently, the case of $a = b = 1 + \epsilon$ missing in our theorem would include the larger GAP class studied by Papadimitriou and Yannakakis [35]. Our current techniques to prove Theorem 13 do not allow us to translate such bi-approximations of BCOs into those for DCPs (at least not without demanding certain stronger condition than in Type 5).
- Can the diversity computational problem for approximate TSP tours or Max-TSP tours be solved in polynomial time?

────── **References** ──────

1 Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi R. Varada-rajan. Diverse near neighbor problem. In *Symposium on Computational Geometry (SoCG)*, pages 207–214. ACM, 2013.

2 Giorgio Ausiello, Alberto Marchetti-Spaccamela, Pierluigi Crescenzi, Giorgio Gambosi, Marco Protasi, and Viggo Kann. *Complexity and approximation: combinatorial optimization problems and their approximability properties.* Springer, 1999.

3 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.

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

5 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Math. Program.*, 128(1-2):355–372, 2011.

6 Benjamin E. Birnbaum and Kenneth J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 55(1):42–59, 2009.

7 Allan Borodin, Aadhar Jain, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions, and dynamic updates. *ACM Trans. Algorithms*, 13(3):41:1–41:25, 2017.

8 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *J. Algorithms*, 13(2):258–273, 1992.

9 Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, 2000.

10 Jens Clausen and Lone Aalekjær Hansen. Finding k edge-disjoint spanning trees of minimum total weight in a network: an application of matroid theory. In *Combinatorial Optimization II*, pages 88–101. Springer, 1980.

11 Clayton W Commander, Panos M Pardalos, Valeriy Ryabchenko, Stan Uryasev, and Grigoriy Zrazhevsky. The wireless network jamming problem. *Journal of Combinatorial Optimization*, 14(4):481–498, 2007.

12 Ilias Diakonikolas and Mihalis Yannakakis. Small approximate pareto sets for biobjective shortest paths and other problems. *SIAM Journal on Computing*, 39(4):1340–1371, 2010.

13 Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

14 Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

15 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)*, volume 181 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

16 Fedor V. Fomin, Petr A. Golovach, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. Diverse collections in matroids and graphs. In *38th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 187 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

17 Harold N Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM Journal on Computing*, 6(1):139–150, 1977.

18 Elazar Goldenberg and Karthik C. S. Hardness amplification of optimization problems. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

19 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. *CoRR*, abs/2201.08940, 2022.

20   Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, See Woo Lee, and Yota Otachi. Computing diverse shortest paths efficiently: A theoretical and experimental study. *CoRR*, abs/2112.05403, 2021.

21   Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, and Yota Otachi. Finding diverse trees, paths, and more. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 3778–3786. AAAI Press, 2021.

22   Satoshi Hara and Takanori Maehara. Enumerate lasso solutions for feature selection. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 1985–1991. AAAI Press, 2017.

23   Refael Hassin, Shlomi Rubinstein, and Arie Tamir. Approximation algorithms for maximum dispersion. *Oper. Res. Lett.*, 21(3):133–137, October 1997.

24   Arne Herzel, Cristina Bazgan, Stefan Ruzika, Clemens Thielen, and Daniel Vanderpooten. One-exact approximate pareto sets. *Journal of Global Optimization*, 80(1):87–115, 2021.

25   Arne Herzel, Stefan Ruzika, and Clemens Thielen. Approximation methods for multiobjective optimization problems: A survey. *INFORMS Journal on Computing*, 33(4):1284–1299, 2021.

26   Jakob Krarup. The peripatetic salesman and some related unsolved problems. In *Combinatorial programming: methods and applications*, pages 173–178. Springer, 1995.

27   Michael J. Kuby. Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. *Geographical Analysis*, 19(4):315–329, October 1987.

28   Eugene L. Lawler. 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, 1972.

29   Erik M Lindgren, Alexandros G Dimakis, and Adam Klivans. Exact map inference by avoiding fractional vertices. In *International Conference on Machine Learning*, pages 2120–2129. PMLR, 2017.

30   Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 954–961. ACM, 2017.

31   Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science*, pages 17–27, 1980.

32   Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987.

33   Katta G Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations research*, 16(3):682–687, 1968.

34   João Carlos Namorado Climaco and Ernesto Queirós Vieira Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4):399–404, 1982.

35   Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st annual symposium on foundations of computer science (FOCS)*, pages 86–92. IEEE, 2000.

36   Ram Ravi and Michel X Goemans. The constrained minimum spanning tree problem. In *Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 66–75. Springer, 1996.

37   Sekharipuram S Ravi, Daniel J Rosenkrantz, and Giri Kumar Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

38   Da-Wei Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information Processing Letters*, 28(6):281–286, 1988.

39   Hao-Tsung Yang, Shih-Yu Tsai, Kin Sum Liu, Shan Lin, and Jie Gao. Patrol scheduling against adversaries with varying attack durations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1179–1188, 2019.