RAIRO-Oper. Res. 57 (2023) 857–880
https://doi.org/10.1051/ro/2023034

RAIRO Operations Research
www.rairo-ro.org

POLYNOMIAL ALGORITHMS FOR *P*-DISPERSION PROBLEMS IN A PLANAR PARETO FRONT

Nicolas Dupin*

Abstract. In this paper, p-dispersion problems are studied to select $p \geqslant 2$ representative points from a large 2D Pareto Front (PF), solution of bi-objective optimization. Four standard p-dispersion variants are considered. A novel variant, Max-Sum-Neighbor p-dispersion, is introduced for the specific case of a 2D PF. Firstly, 2-dispersion and 3-dispersion problems are proven solvable in $O(n)$ time in a 2D PF. Secondly, dynamic programming algorithms are designed for three p-dispersion variants, proving polynomial complexities in a 2D PF. Max-min p-dispersion is solvable in $O(pn \log n)$ time and $O(n)$ memory space. Max-Sum-Neighbor p-dispersion is proven solvable in $O(pn^2)$ time and $O(n)$ space. Max-Sum-min p-dispersion is solvable in $O(pn^3)$ time and $O(pn^2)$ space. These complexity results hold also in 1D, proving for the first time that Max-Sum-min p-dispersion is polynomial in 1D. Furthermore, properties of these algorithms are discussed for an efficient implementation and for practical applications.

Mathematics Subject Classification. 90C39, 90C23, 90B80, 90-08.

Received September 24, 2021. Accepted March 22, 2023.

1. Introduction

In real-world applications, optimization problems may be driven by several conflicting objectives. Designing network or (system of) system architectures, financial costs must be traded off with quality of service or robustness [12, 43]. Dealing with complex maintenance planning problems, stability and robustness of the planning matter as well as financial costs [15]. Multi-objective optimization (MOO) supports such decision making. Many efficient (*i.e.* best compromise) solutions of MOO problems may exist with Pareto dominance [20]. A Pareto Front (PF) denotes the projection of efficient solutions in the objective space. This work aims to select p solutions from $n \gg p$ non dominated solutions, while maximizing the diversity of these p solutions in the objective space. Firstly, such problem occurs when selecting alternatives for decision makers. Secondly, MOO approaches use such operators to represent large PF [4, 46], and MOO meta-heuristics archive diversified solutions during the heuristic search [49, 52, 57]. Thirdly, a similar problem occurs in a Skyline for databases [5, 8, 36]. When selecting alternatives for decision makers, p is small, $p \leqslant 5$ is realistic. Otherwise, p is larger, having $p = 100$ or $p = 1000$ is realistic.

Keywords. Optimization, Algorithms, Dynamic programming, p-dispersion, Complexity, Bi-objective optimization, Pareto Front, Skyline operator.

Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France.
* Corresponding author: nicolas.dupin@univ-angers.fr

The hypervolume measure is often used in such a context [3, 24, 29]. Covering and clustering algorithms [17, 19, 57] are also used to select points in a PF. In this paper, we consider (discrete) p-dispersion problems, to select p points and while maximizing dispersion measures among selected points [22]. Although p-dispersion is mentioned to have relevant applications for MOO [23, 45], no specific studies concerned the p-dispersion in a PF to the best of our knowledge. Four variants of discrete p-dispersion problems are defined [23]. Max-min and Max-Sum p-dispersion problems, the most studied variants, are $\mathcal{NP}$-hard [22, 30]. Max-min p-dispersion maximizes the minimal distance between each pair of selected points. Max-Sum p-dispersion maximizes the total sum of the distances between selected points. Max-Sum-min p-dispersion variant maximizes the sum of the distances between each selected points to the closest different selected points. Max-min-Sum p-dispersion variant maximizes the minimal sum of distances between each selected point to the other selected points. This paper studies p-dispersion variants in the case of a two-dimensional (2D) PF, which is an extension of one-dimensional (1D) cases. A novel variant of Max-Sum-min p-dispersion, denoted Max-Sum-Neighbor p-dispersion, is specifically introduced for 2D PFs. For these five p-dispersion problems, the cases $p = 2$ and $p = 3$ are proven to be solvable in $O(n)$ time and the cases $p = 4$ and $p = 5$ are solvable respectively in $O(n^2)$ and $O(n^3)$ time. Generally, the Max-min, Max-Sum-min and Max-Sum-Neighbor p-dispersion problems are proven to be solvable in polynomial time in a 2D PF with dynamic programming (DP) algorithms. For Max-Sum-min p-dispersion, it is the first time that this problem is proven to be polynomially solvable in 1D.

The remainder of this paper is organized as follows. In Section 2, we introduce the notation and formally describe the problems. In Section 3, we discuss related state-of-the-art elements to situate our contributions. In Section 4, intermediate results are presented. In Sections 5–7, DP algorithms with a polynomial complexity are respectively presented for the Max-Sum-Neighbor, Max-min and Max-min-Sum variants. In Section 8, our results are discussed from both a theoretical and a practical point of view. In Section 9, our contributions and the open perspectives are summarized.

2. Problem statement

Let $E = \{x_1, \dots, x_n\}$ be a set of n points in a 2D PF, considering the minimization of two objectives (this is not a loss of generality). We denote discrete intervals $[\![a, b]\!] = [a, b] \cap \mathbb{Z}$, so that we can use the notation of discrete index sets and write $E = \{x_i\}_{i \in [\![1,n]\!]}$. As in [17, 19], PFs are formally defined using binary relations: relation $\mathcal{I}$ expresses Pareto incomparability, whereas relation $\prec$ defines an order from left to right, as illustrated in Figure 1. Binary relations $\mathcal{I}, \prec$ are defined for all $y = (y^1, y^2), z = (z^1, z^2) \in \mathbb{R}^2$ with:

$$y \prec z \iff y^1 < z^1 \text{ and } y^2 > z^2 \tag{1}$$

$$y \preccurlyeq z \iff y \prec z \text{ or } y = z \tag{2}$$

$$y \ \mathcal{I} \ z \iff y \prec z \text{ or } z \prec y. \tag{3}$$

A 2D PF can be the projected costs of efficient solutions using exact approaches in discrete MOO problems [20], or using population meta-heuristics like an evolutionary algorithm (EA) [52]. In the context of databases, Skyline operators are also PFs [5]. A 2D PF can be extracted from any subset of $\mathbb{R}^2$ using an output-sensitive algorithm [42]. MOO problems with continuous variables may have as solution a continuous PF. We will discuss in Section 8.4 how to use the results of this paper in this last case. Finally, an affine 2D PF is similar to a 1D instance, we will formalize it.

A strong assumption in this paper is that the 2D PF E is known a priori: p-dispersion problems are computed knowing E. This version of the problem is denoted *explicit*, unlike the so-called *implicit* versions where points of the PF are selected by simultaneously calculating the PF. In the context of MOO optimization, *implicit* p-dispersion would be combined with MOO approaches as in [4]. Context of databases are less structured: an *implicit* version of p-dispersion would consider the total number of rows in the database $h \gg n$ that can be enumerated where n is the size of the Skyline, which is unknown at the beginning of the search, as in [8].

The distance between points $x_i, x_j \in E$ is denoted $d_{ij} = d(x_i, x_j)^\alpha$ where $\alpha > 0$ and $d(y, z)$ denotes a Chebyshev or a Minkowski distance, induced by the ℓ_∞ and ℓ_m norms. For a given $m > 0$, Minkowski distance

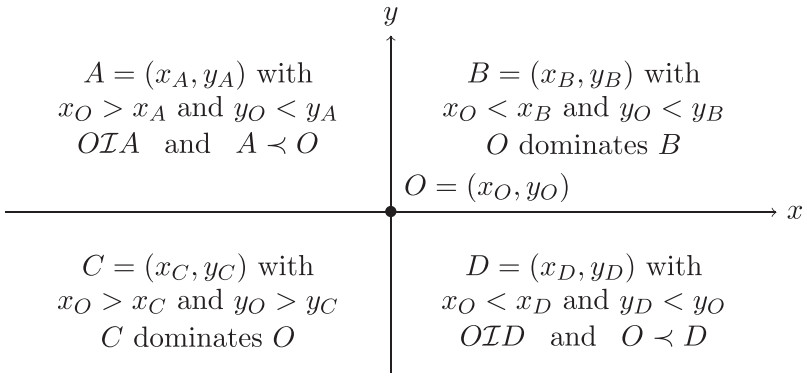

FIGURE 1. Illustration of relations $\mathcal{I}, \prec$ and Pareto dominance minimizing two objectives indexed by x and y.

d_m is defined by the following formula for $y = (y^1, y^2),\ z = (z^1, z^2) \in \mathbb{R}^2$:

$$\forall y, z \in \mathbb{R}^2, \qquad d_m(y,z) = \sqrt[m]{|y^1 - z^1|^m + |y^2 - z^2|^m}. \tag{4}$$

The case $m = 2$ corresponds to the Euclidean distance. The limit with $m \to \infty$ defines the Chebyshev distance, denoted d_∞:

$$\forall y, z \in \mathbb{R}^2, \qquad d_\infty(y,z) = \max\big(|y^1 - z^1|, |y^2 - z^2|\big). \tag{5}$$

In p-dispersion problems, the task is to select $p \geqslant 2$ out of n given candidate points, while maximizing a dispersion function f:

$$\mathcal{P}_{\text{disp}}(E,p) = \max_{(z_1,z_2,\ldots,z_p)\in D_p} f(z_1, z_2, \ldots, z_p), \tag{6}$$

where D_p denotes the set of all the p-tuples with distinct points of E:

$$D_p = \{(z_1, z_2, \ldots, z_p) \in E^p \mid \forall 1 \leqslant i < j \leqslant p, z_i \neq z_j\}. \tag{7}$$

The most standard p-dispersion problem is also denoted Max-min p-dispersion problem or p-dispersion-Mm. The dispersion function is in this case the minimum of distances d_{ij} between pairs of the selected points. The Max-min p-dispersion problem for $p \geqslant 2$ is written as:

$$\mathcal{P}^{\text{Mm}}_{\text{disp}}(E,p) = \max_{(z_1,z_2,\ldots,z_p)\in D_p} \min_{i,j:1\leqslant i<j\leqslant p} d_{ij}. \tag{8}$$

Max-Sum(-Sum) dispersion problem, denoted p-dispersion-MS, considers as dispersion function the total sum of distances among selected points:

$$\mathcal{P}^{\text{MS}}_{\text{disp}}(E,p) = \max_{(z_1,z_2,\ldots,z_p)\in D_p} \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} d_{ij}. \tag{9}$$

We consider also the Max-Sum-min p-dispersion variant, denoted p-dispersion-MSm [23], where dispersion is measured as the sum of the distance of each selected points to its closest (and different) selected point:

$$\mathcal{P}^{\text{MSm}}_{\text{disp}}(E,p) = \max_{(z_1,z_2,\ldots,z_p)\in D_p} \sum_{i=1}^{p} \min_{j\in[\![1,p]\!]-\{i\}} d_{ij}. \tag{10}$$

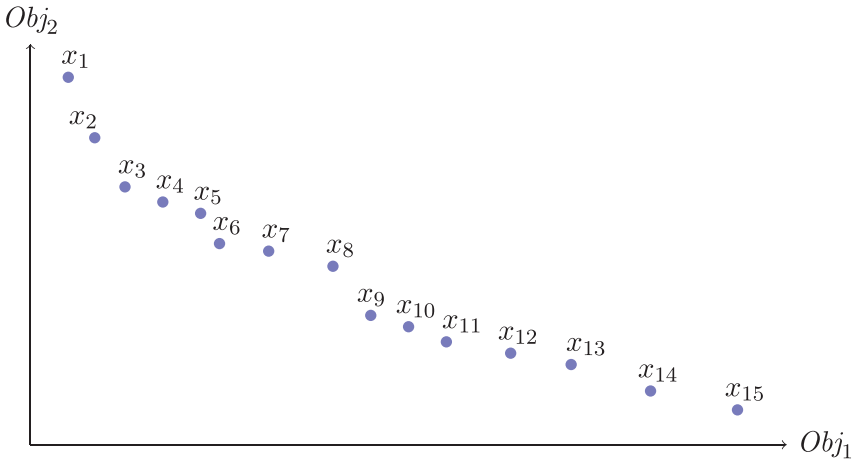

FIGURE 2. Illustration of a 2D PF with 15 points and the indexing implied by Lemma 1

The Max-min-Sum p-dispersion problem [23], denoted p-dispersion-MmS, is defined with a dispersion calculated as the sum of distances of each selected point to the other selected points:

$$\mathcal{P}_{\text{disp}}^{\text{MmS}}(E,p) = \max_{(z_1,z_2,\dots,z_p)\in D_p} \min_{i\in[\![1,p]\!]} \sum_{j\in[\![1,p]\!]-\{i\}} d_{ij}. \tag{11}$$

Note that 1D instances are special cases of 2D PF, equivalent to aligned points in a 2D PF. Considering any variant of p-dispersion problems, only the relative distance matters. Hence, p-dispersion problems for aligned points in the plane are equivalent to 1D instances and in these cases, Minkowski and Chebyshev distances are the same.

The previous definitions are generic and do not use specificities of 2D PFs. Lemmas 1 and 2, mentioned and proven in [19], allows to reformulate the definitions of some p-dispersion variants in a 2D PF, as well as defining a new variant that will be specific for 2D PFs:

Lemma 1. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a 2D PF. The relation $\preccurlyeq$ is an order and $\prec$ is transitive. E can be re-indexed in $O(n\log n)$ time such that:*

$$\forall (i_1,i_2)\in[\![1,n]\!]^2, \qquad i_1 < i_2 \Longrightarrow x_{i_1} \prec x_{i_2}, \tag{12}$$
$$\forall (i_1,i_2)\in[\![1,n]\!]^2, \qquad i_1 \leqslant i_2 \Longrightarrow x_{i_1} \preccurlyeq x_{i_2}. \tag{13}$$

Lemma 2. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF using Lemma 1. The following monotony relations are valid considering a Minkowski or Chebyshev distance d, and any real number $\alpha > 0$:*

$$\forall (i_1,i_2,i_3)\in[\![1,n]\!]^3, \qquad i_1 \leqslant i_2 < i_3 \Longrightarrow d(x_{i_1},x_{i_2})^\alpha < d(x_{i_1},x_{i_3})^\alpha, \tag{14}$$
$$\forall (i_1,i_2,i_3)\in[\![1,n]\!]^3, \qquad i_1 < i_2 \leqslant i_3 \Longrightarrow d(x_{i_2},x_{i_3})^\alpha < d(x_{i_1},x_{i_3})^\alpha. \tag{15}$$

Lemma 1 defines a 1D structure and a total order in a 2D PF. The re-indexing in Lemma 1 is equivalent to a lexicographic sort minimizing hierarchically the two objectives, thus running in$O(n\log n)$ time. We will not use Lemma 1 when the $O(n\log n)$ time complexity degrade the overall complexity of some algorithms. When the indexing of a 2D PF fulfill equations (12), it will be mentioned as a *re-indexed 2D PF*. Without additional precision, a 2D PF will not be considered as re-indexed with Lemma 1.

Using Lemma 2, Max-min and Max-Sum-min p-dispersion problems can be reformulated in a 2D PF considering only consecutive distances:

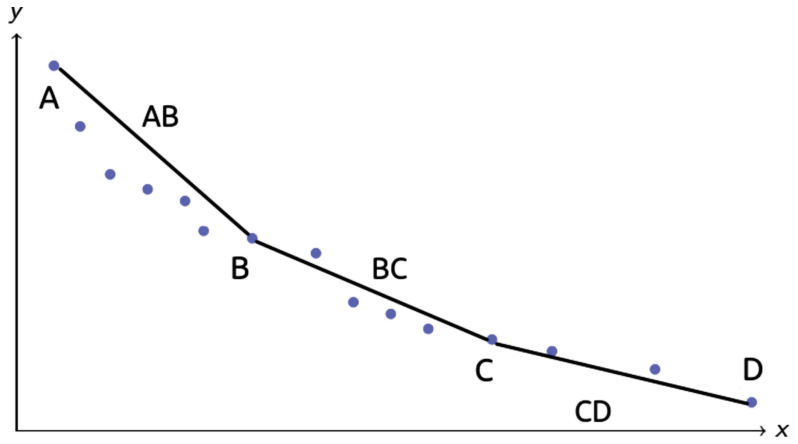

FIGURE 3. Illustration of the different dispersion variants: we consider the 4-dispersion problems, with four selected points, A, B, C and D such that $A \prec B \prec C \prec D$.

Lemma 3. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. The Max-min and Max-Sum-min p-dispersion problems in E are also defined as:*

$$\mathcal{P}_{\text{disp}}^{\text{Mm}}(E,p) = \max_{1\leqslant i_1<\cdots<i_p\leqslant p} \min_{j\in[\![1;p-1]\!]} d_{i_j,i_{j+1}}, \tag{16}$$

$$\mathcal{P}_{\text{disp}}^{\text{MSm}}(E,p) = \max_{1\leqslant i_1<\cdots<i_p\leqslant p} \sum_{j=2}^{p-1} \min(d_{i_j,i_{j+1}}, d_{i_j,i_{j-1}}) + d_{i_1,i_2} + d_{i_{p-1},i_p}. \tag{17}$$

Proof. In the inner minimization of (8) and (10), distances $d_{i_j,i_{j'}}$ are considered. Such distances are higher than $d_{i_j,i_{j+1}}$ and $d_{i_{j'-1},i_{j'}}$ using Lemma 2, and it remains only distances among consecutive points. For the two extreme points x_1 and x_p, it remains only d_{i_1,i_2} and d_{i_{p-1},i_p}. □

Furthermore, Lemma 1 allows to define a new variant of Max-Sum-min p-dispersion, the Max-Sum-Neighbor p-dispersion problem:

Definition 1 (Max-Sum-Neighbor p-dispersion)**.** Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. Max-Sum-Neighbor p-dispersion, denoted p-dispersion-MSN, is defined in E summing only the distances between neighbor points:

$$\mathcal{P}_{\text{disp}}^{\text{MSN}}(E,p) = \max_{1\leqslant i_1<i_2<\cdots<i_p\leqslant p} \sum_{j=1}^{p-1} d_{i_j,i_{j+1}} \tag{18}$$

Dispersion variants are illustrated in Figure 3. Lemma 3 shows that p-dispersion-MSm is not a symmetric expression of distances, inducing more importance to extreme distances AB and CD. On the contrary, p-dispersion-MSN induces symmetrical expressions.

3. Related works

3.1. Complexity results for p-dispersion problems

Max-min and Max-Sum p-dispersion problems are $\mathcal{NP}$-hard for general metric spaces. This is proven in both cases by a polynomial reduction from the maximum independent set problem [22,30]. Max-min and Max-Sum p-dispersion problems are still $\mathcal{NP}$-hard problems when distances fulfill the triangle inequality [22,30].

The planar Max-min p-dispersion problem is also $\mathcal{NP}$-hard [56], the $\mathcal{NP}$-hardness of the planar Max-Sum p-dispersion problem is still an open question to our knowledge.

Approximability and non approximability with constant factors were studied for p-dispersion problems. Unless $\mathcal{P} = \mathcal{NP}$, Max-min p-dispersion cannot be approximated with a constant factor [45]. For general metric spaces, Max-min and Max-Sum p-dispersion problems can be approximated with a $1/2$ factor [53, 54]. For 2D spaces, Max-min and Max-Sum p-dispersion problems can be approximated with a constant factor [45]. The $1/2$ factor is the tightest approximation factor for Max-min p-dispersion with triangle inequality, unless $\mathcal{P} = \mathcal{NP}$ [45].

Some sub-cases of p-dispersion problems are solvable in polynomial time. The 1D cases of Max-min and Max-Sum p-dispersion problems are solvable in polynomial time, with DP algorithms running in $O(\max\{pn, n \log n\})$ time [45, 56]. Within a tree structure, Max-min p-dispersion is solvable in $O(n^2 \log n)$ time [10]. Although, p-dispersion is mentioned to have relevant applications for MOO [23, 45], no specific studies concerned p-dispersion in a PF besides 1D cases to the best of our knowledge. We note an interest for implicit versions of dispersion problems in of Skyline Operators [36, 37, 55].

3.2. Exact methods for p-dispersion problems

Max-Sum p-dispersion can be formulated as a quadratic optimization problem, defining binary variables $z_j \in \{0, 1\}$ with $z_j = 1$ if and only if the point x_j is selected:

$$\begin{array}{rlll} \mathcal{P}^{\mathrm{MS}}_{\mathrm{disp}}(E, p) = \max & \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{i,j} z_i z_j & & (19.1) \\ \text{s.t.:} & \sum_{j=1}^{n} z_j = p & & (19.2) \\ & z_j \in \{0, 1\} & \forall j \in [\![1, n]\!]. & (19.3) \end{array} \tag{19}$$

Linearizing (19.1) leads to the Integer Linear Programming (ILP) formulation provided in [33]. Exact Branch&Bound (B&B) algorithms were also provided computing iteratively higher and lower bounds, with a Lagrangian relaxation [2] or with tailored higher bounds computable in $O(n^3)$ time [44].

Max-min p-dispersion is also a non linear optimization problem [44]:

$$\begin{array}{rlll} \mathcal{P}^{\mathrm{Mm}}_{\mathrm{disp}}(E, p) = & \max_{d \geqslant 0} d & & (20.1) \\ \text{s.t:} & \sum_{j=1}^{n} z_j = p & & (20.2) \\ & d z_i z_j \leqslant d_{i,j} & \forall 1 \leqslant i < j \leqslant n, & (20.3) \\ & z_j \in \{0, 1\} & \forall j \in [\![1, n]\!]. & (20.4) \end{array} \tag{20}$$

The standard linearization of constraints (20.3) leads to the Mixed Integer Linear Programming (MILP) formulation [33]. An alternative MILP formulation and specific cuts for a Branch&Cut algorithm is provided by [47]. Decomposition schemes from [2] and [44] have been also extended for the Max-min p-dispersion problem.

Similarly, MILP formulations were designed for the Max-Sum-min and Max-min-Sum p-dispersion variants [23]. Such variants were less studied. A recent work proposed a unified MILP formulation and B&B algorithm for the four variants of p-dispersion problems [35].

3.3. Clustering/selecting points in PFs

We summarize here results related to the explicit versions of selection or clustering of points in a PF.

Maximizing the quality of discrete representations of Pareto sets was studied with hypervolume measure in the Hypervolume Subset Selection (HSS) problem [3, 46]. The HSS problem, maximizing representativeness of k solutions among a PF of size n, is $\mathcal{NP}$-hard in 3D (and higher dimensions) [6]. An exact algorithm in $n^{O(\sqrt{k})}$ in 3D and a polynomial-time approximation scheme for any constant dimension d were also provided [6]. 2D PF instances are solvable in polynomial time, a DP algorithm running in $O(kn^2)$ time and using $O(kn)$ space was firstly provided in [3]. The time complexity was improved in $O(kn + n \log n)$ by [7] and in $O(k(n-k) + n \log n)$ by [34]. Some similar results exist also for clustering problems. The k-median and k-medoid problems are $\mathcal{NP}$ hard in 2D since [40]. The 2D PF cases are solvable in $O(n^3)$ time with DP algorithms [16, 17]. The 1D cases are solvable in $O(nk)$ time [31]. Min Sum of Square Clustering (MSSC), also denoted k-means problem, is also

$\mathcal{NP}$-hard for 2D cases [38]. The 1D cases of k-means are also solvable by a DP algorithm, with a complexity in $O(kn)$ using memory space in $O(n)$ [28].

Similar results are available for variants of p-center problems. The p-center problems minimize the radius of a ball, to cover all the points with p such identical balls. Contrary to the continuous version, discrete p-center variant consider the additional constraint to have a discrete set of points as candidates for ball centers. It makes sense to consider the original points as such candidates for centers, as in [19]. The dual of a Max-min p-dispersion is similar to a min-max optimization as in p-center problems. Duality relations hold between p-dispersion-Mm and p-center problems in 1D and in tree structures: p-dispersion-Mm is the dual of the continuous $(p-1)$-center for such cases [50]. Max-min p-dispersion and p-center problems have similar complexity results. The discrete and continuous p-center problems are $\mathcal{NP}$-hard in general, the discrete p-center problem in $\mathbb{R}^2$ with a Euclidean distance is also $\mathcal{NP}$-hard [40]. The 1D and 2D PF sub-cases of p-center problems are polynomially solvable with DP algorithms. The 1D continuous p-center is solvable in $O(n \log^3 n)$ time [41], whereas the time complexity in a 2D PF is in $O(pn \log n)$ [18,19]. The discrete p-center problem in a 2D PF is solvable in $O(n \log n)$ time and $O(n)$ space [8], whereas it is solvable in $O(n)$ time in 1D [25]. The Min-sum p-radii problems, denoted also min-sum-diameter, are p-center variants, where the covering balls may not be identical. It is a min-Sum-Max optimization, that we can compare to the Max-Sum-min optimization with p-dispersion-MSm. Min-sum p-radii is $\mathcal{NP}$-hard in the general case and polynomial within a tree structure [13]. The $\mathcal{NP}$-hardness is also proven even in metrics induced by weighted planar graphs [26]. In a 2D PF, Min-sum p-radii is solvable in $O(pn^2)$ time with a DP algorithm [19]. In 1D, a specific algorithm runs in $O(n \log n)$ time [19].

Lastly, partial variants were studied for p-center problems and variants in a 2D PF, allowing that m points are not covered. These points can be outliers to remove or isolated points that are wished to be detected in the context of EAs [19]. DP algorithms for 2D PF can be extended allowing to uncover $m > 0$ points. The time and space complexity of DP algorithms are then multiplied by a factor m [19]. In particular, partial p-center problems in a 2D PF are solvable in $O(mpn \log n)$ time and $O(mn)$ space [19].

3.4. Summary of contributions and relations to state of the art

From this literature review, some common results appear. General 2D instances of the clustering and selection problems are often $\mathcal{NP}$-hard. DP algorithms induce a polynomial complexity in 1D and some 2D PF cases. Many DP algorithms use a $p \times n$ matrix to store results of $k \leqslant p$ selection/clustering among the $n' \leqslant n$ first points. This induces $p \times n$ computations of a partial solution using $O(n)$ previous ones. A linear enumeration induces time complexities in $O(pn^2)$ whereas some $O(pn \log n)$ time complexities can be obtained using a logarithmic search. Our DP algorithms for p-dispersion-MSN and p-dispersion-Mm have this form. Other techniques make it possible to divide the complexity by a factor of p, as in [8,28]. It is an open perspective for the DP algorithms provided in this paper.

Table 1 summarizes the complexity results for selection and clustering problems for the 1D, 2D PF and 2D sub-cases. A first comparison between these problems situates the selection problems p-center, p-dispersion-Mm and HSS as the fastest to solve in 2D PFs. Clustering with p-median or p-medoids is more accurate for measuring cluster similarity, however the time complexity in $O(n^3)$ is a burden for application. Time complexity in $O(pn^3)$ for p-dispersion-MSm is a more a theoretical than a practical result. Surprisingly, p-dispersion-MSN, which seems to be a slight variant of p-dispersion-MSm, has a much better complexity. Min-sum p-radii and p-dispersion-MSN have the same time complexity in $O(pn^2)$ with similar DP algorithms. Both variants improve weaknesses of max-min and min-max optimization with p-center and p-dispersion-Mm, such as the possible large number of optimal solutions, including solutions potentially very unbalanced, to obtain fewer and better balanced solutions. Such weaknesses of p-dispersion-Mm are highlighted in Proposition 10, similar algorithms and results for p-center problems are described in [19].

Complexity results can be significantly worse for 2D PFs than the 1D sub-cases, as for the Min-sum p-radii and p-medoids. In such cases, triangle inequality instead of additivity of distances is crucial. For p-center and p-dispersion problems, the time complexity for 2D PFs is not significantly worse than for 1D sub-cases. Within a tree structure, which is another extension of 1D cases, p-dispersion-Mm is solvable in $O(n^2 \log n)$ time [10]

TABLE 1. Comparison of the time complexity obtained after our study for dispersion problems (in bold) with related problems on 1D, 2D PF and 2D cases and their reference. Some cases are still open questions. BF denotes brute force naive enumeration. Note that p-dispersion-MSN is defined only for 1D and 2D PF instances, not for general 2D instances

Problem	1D		2D PF		2D	
p-dispersion-Mm	$O(n(p+\log n))$	[45]	$\mathbf{O(pn\log n)}$		$\mathcal{NP}$-hard	[56]
p-dispersion-MSN	$\mathbf{O(pn^2)}$		$\mathbf{O(pn^2)}$		not defined	
p-dispersion-MSm	$\mathbf{O(pn^3)}$		$\mathbf{O(pn^3)}$		open	
p-dispersion-MS	$O(n(p+\log n))$	[56]	open		open	
2-dispersion	$\mathbf{O(n)}$		$\mathbf{O(n)}$		$O(n^2)$	BF
3-dispersion	$\mathbf{O(n)}$		$\mathbf{O(n)}$		$O(n^3)$	BF
4-dispersion	$\mathbf{O(n^2)}$		$\mathbf{O(n^2)}$		$O(n^4)$	BF
Cont. p-center	$O(n\log^3 n)$	[41]	$O(pn\log n)$	[19]	$\mathcal{NP}$-hard	[40]
Discr. p-center	$O(n)$	[25]	$O(n\log n)$	[8]	$\mathcal{NP}$-hard	[40]
Min-sum p-radii	$O(n\log n)$	[19]	$O(pn^2)$	[19]	$\mathcal{NP}$-hard	[26]
Cont. 2-center	$O(n\log n)$	[19]	$O(n\log n)$	[19]	$O(n\log^2 n)$	[39]
Discr. 2-center	$O(n\log n)$	[19]	$O(n\log n)$	[19]	$O(n^{4/3}\log^5 n)$	[1]
HSS	undefined		$O(n(p+\log n))$	[34]	$\mathcal{NP}$-hard	[6]
p-means	$O(pn)$	[28]	open		$\mathcal{NP}$-hard	[38]
p-median	$O(pn)$	[31]	$O(n^3)$	[16].	$\mathcal{NP}$-hard	[40]
p-medoids	$O(pn)$	[31]	$O(n^3)$	[17].	$\mathcal{NP}$-hard	[40]

instead of $O(pn\log n)$ in 2D PF, the time complexity in a 2D PF is significantly better. Note that Max-Sum-min p-dispersion was not studied before in 1D to the best of our knowledge, so that Theorem 3 and Corollary 1 proves for the first time that Max-Sum-min p-dispersion in 1D is solvable in polynomial time. Perspectives may be to improve this complexity result using distance additivity in 1D.

4. INTERMEDIATE RESULTS

This section presents intermediate results that will be a basis for future developments. A key element is that the extreme points of a re-indexed 2D PF are natural candidates for p-dispersion problems:

Proposition 1 (2-dispersion problems)**.** *Let $E=\{x_i\}_{i\in[\![1,n]\!]}$ be a 2D PF. 2-dispersion problems are solvable in $O(n)$ time using $O(1)$ additional memory space in the 2D PF E, considering any variant of p-dispersion. There is a unique optimal solution, selecting the extreme points x_1 and x_n.*

Proof. Any 2-dispersion variant consider the same problem:

$$\mathcal{P}_2(E)=\max_{1\leqslant i<j\leqslant p} d(x_i,x_j). \tag{21}$$

Indeed, $\mathcal{P}^{\mathrm{Mm}}_{\mathrm{disp}}(E,2)=\mathcal{P}^{\mathrm{MS}}_{\mathrm{disp}}(E,2)=\mathcal{P}^{\mathrm{MmS}}_{\mathrm{disp}}(E,2)=\mathcal{P}^{\mathrm{MSN}}_{\mathrm{disp}}(E,2)=\mathcal{P}_2(E)$ and $\mathcal{P}^{\mathrm{MSm}}_{\mathrm{disp}}(E,2)=2\mathcal{P}_2(E)$. Using Lemma 2, $\mathcal{P}_2(E)=d_{1,n}$, selecting the two extreme points x_1 and x_n after re-indexing. The complexity, once having computed x_1 and x_n is in $O(1)$ time and additional space. Re-indexing E induces a complexity in $O(n\log n)$ time. By computing only extreme points with a single traversal of E, time complexity is in $O(n)$. □

Proposition 2 (p-dispersion and extreme points)**.** *Let $E=\{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. For each p-dispersion variant, an optimal solution exists selecting x_1 and x_n for $p\geqslant 2$. Reciprocally, any optimal solution contains the extreme points in the case of the Max-Sum and Max-Sum-Neighbor variants.*

Proof. Let $1 \leqslant i_1 < i_2 < \cdots < i_p \leqslant p$ be the indexes defining an optimal solution of a p-dispersion variant. Considering new indexes $i'_1 = 1,\ i'_2 = i_2, \ldots i'_{p-1} = i_{p-1}, i'_p = p$, Lemma 2 implies that $d(x_{i_j}, x_{i_{j'}})^\alpha \leqslant d(x_{i'_j}, x_{i'_{j'}})^\alpha$ for all $j, j' \in [\![1, p]\!]$. Points $x_{i'_1}, \ldots, x_{i'_{p-1}}, x_{i'_p}$ have at least the same dispersion than the original points $x_{i_1}, \ldots, x_{i_{p-1}}, x_{i_p}$. Hence, it defines an optimal solution of the considered p-dispersion variant. Having $1 < i_1$ or $i_p < p$, Lemma 2 induces $d(x_{i_1}, x_{i_2})^\alpha + d(x_{i_{p-1}}, x_{i_p})^\alpha < d(x_{i'_1}, x_{i'_2})^\alpha + d(x_{i'_{p-1}}, x_{i'_p})^\alpha$. Points $x_{i'_1}, \ldots, x_{i'_{p-1}}, x_{i'_p}$ would induce a strictly better solution than an optimal solution of Max-Sum or Max-Sum-Neighbor p-dispersion. By contradiction, the extreme points are in any optimal solution for both variants. □

Remark 1. For Max-min p-dispersion, optimal solutions exist without containing x_1 and x_n. We consider 3-dispersion-Mm and 4 points $x_1 = (0, 10)$, $x_2 = (1, 9)$, $x_3 = (3, 7)$, $x_4 = (5, 5)$. Using the Euclidean distance, $d(x_1, x_2) = \sqrt{2}$, $d(x_2, x_3) = 2\sqrt{2}$, $d(x_1, x_3) = 3\sqrt{2}$, $d(x_3, x_4) = 2\sqrt{2}$. Hence, $\{x_2, x_3, x_4\}$ has the same dispersion-Mm as $\{x_1, x_3, x_4\}$, $2\sqrt{2}$, which is optimal.

Remark 2. Clustering measure like k-means, k-medoids or k-center variants do not return the extreme points in general.

Proposition 3 (3-dispersion)**.** *3-dispersion problems in a 2D PF are solvable in $O(n)$ time using $O(1)$ additional space.*

Proof. Considering variants (8)–(11) or (18), we consider the two extreme points, which can be found in $O(n)$ time with one traversal of E. Then, there are $n - 2$ cases to enumerate the last point, each cost computation of 3-dispersion being in $O(1)$ time, this last naive enumeration is in $O(n)$ time. 3-dispersion problems are thus solved in $O(n)$ time using $O(1)$ additional space with two traversals of E. □

Proposition 4. *In a 2D PF, the p-dispersion problems are solvable in $O\left(p^2 \binom{n-2}{p-2}\right)$ time using $O(1)$ additional space.*

Proof. Similarly to Proposition 3, once extreme points are computed in $O(n)$ time, the naive enumeration of other $p - 2$ selected points induces $\binom{n-2}{p-2}$ computations, requiring $O(p)$ or $O(p^2)$ time computations. □

Remark 3. With $p \ll n$, the time complexity is roughly in $O(n^{p-2})$, instead of $O(n^p)$ for the naive enumeration. Using Proposition 4, cases $p = 4, 5, 6$ have respectively a time complexity in $O(n^2)$, $O(n^3)$ and $O(n^4)$.

A specific result holds for p-dispersion-MSN in 1D. Proposition 5 and its proof show that it makes sense to consider this problem only for $\alpha \neq 1$:

Proposition 5. *Let $E = \{x_i\}_{i \in [\![1,n]\!]}$ be a set of n distinct real numbers, let $p \geqslant 2$. If $\alpha = 1$, p-dispersion-MSN problem is solvable in $O(n)$ time.*

Proof. Using Proposition 2, one selects the two extreme points. Let $a = \min E$ and $b = \max E$, a and b are computed in $O(n)$ time. With $\alpha = 1$, adding any subset of size $p - 2$ of distinct elements of $E \setminus \{a, b\}$, we will have the same MSN dispersion of $b - a > 0$ because of the distance additivity in 1D, which is trivially optimal. □

5. p-DISPERSION-MSN IS POLYNOMIALLY SOLVABLE IN A 2D PF

Lemma 2 implies Bellman equations for p-dispersion-MSN:

Proposition 6. *Let $E = \{x_i\}_{i \in [\![1,n]\!]}$ be a re-indexed 2D PF. Defining $C^{\text{MSN}}_{k,i}$ as the optimal cost of k-dispersion-MSN among the points re-indexed in $[\![1, i]\!]$ for all $k \in [\![2, p]\!]$ and $i \in [\![k, n]\!]$, we have:*

$$\forall i \in [\![1, n]\!], \qquad C^{\text{MSN}}_{2,i} = d_{1,i} \tag{22}$$

$$\forall k \in [\![3, p]\!],\ \forall i \in [\![k, n]\!], \qquad C^{\text{MSN}}_{k,i} = \max_{j \in [\![k-1, i-1]\!]} \left(C^{\text{MSN}}_{k-1,j} + d_{j,i}\right). \tag{23}$$

Proof. Equation (22) is given by Proposition 1. We suppose $k \geqslant 3$ and prove (23). Let $i \in [\![k, n]\!]$. Selecting for each $j \in [\![k-1, i-1]\!]$ an optimal solution of $(k-1)$-dispersion-MSN among points indexed in $[\![1, j]\!]$, and adding point i, it defines a feasible solution for k-dispersion-MSN among points indexed in $[\![1, i]\!]$ with a cost $C^{\text{MSN}}_{k-1,j} + d_{j,i}$. This last cost is lower than the optimal k dispersion cost, thus $C^{\text{MSN}}_{k,i} \geqslant C^{\text{MSN}}_{k-1,j} + d_{j,i}$. Therefore

$$C^{\text{MSN}}_{k,i} \geqslant \max_{j \in [\![k-1, i-1]\!]} \left(C^{\text{MSN}}_{k-1,j} + d_{j,i}\right). \tag{24}$$

Let $j_1 < j_2 < \cdots < j_{k-1} < j_k$ be indexes defining an optimal solution of k-dispersion-MSN, its cost is $C^{\text{MSN}}_{k,i}$. Because of Proposition 2, we can assume that $j_1 = 1$ and $j_k = i$. Necessarily, $j_1, j_2, \ldots, j_{k-1}$ defines an optimal solution of $(k-1)$-dispersion-MSN among points indexed in $[\![1, j_{k-1}]\!]$. On the contrary, a strictly better solution for $C^{\text{MSN}}_{k,i}$ would be constructed adding the index i. We have thus: $C^{\text{MSN}}_{k,i} = C^{\text{MSN}}_{k-1,j_{k-1}} + d_{j_{k-1},i}$. Combined with (24), it proves : $C^{\text{MSN}}_{k,i} = \max_{j \in [\![k-1, i-1]\!]} \left(C^{\text{MSN}}_{k-1,j} + d_{j,i}\right)$. □

Algorithm 1 is a first DP algorithm for p-dispersion-MSN based on Proposition 6. The first phase computes the matrix of optimal costs $C^{\text{MSN}}_{k,i}$ with index k increasing. $C^{\text{MSN}}_{p,n}$ is the optimal value of MSN p-dispersion. Then, backtracking operations in the matrix $C^{\text{MSN}}_{k,i}$ return an optimal solution.

Algorithm 1. p-dispersion-MSN in a 2D PF with $p \geqslant 3$.

Input: n points of a 2D PF, $E = \{x_i\}_{i \in [\![1,n]\!]}$; an integer $p \in [\![3, n]\!]$.
re-index E following the order of Lemma 1
initialize matrix C with $C_{k,i} := 0$ for all $i \in [\![1, n]\!], k \in [\![2; p-1]\!]$
for $i = 1$ to $N - 1$:
    $C_{2,i} := d_{1,i}$
end for
for $k = 3$ to $p - 1$:
    for $i = k$ to $n - 1$:
        $C_{k,i} := \max_{j \in [\![k-1, i-1]\!]} (C_{k-1,j} + d_{j,i})$
    end for
end for
$j := \text{argmax}_{j \in [\![p-1, N-1]\!]} (C_{p-1,j} + d_{j,N})$; OPT $:= C_{p-1,j} + d_{j,N}$
initialize $i := j$ and $\mathcal{J} := \{1, j, N\}$.
for $k = p - 1$ to 3 with increment $k \leftarrow k - 1$:
    $j := \text{argmax}_{j' \in [\![k-1, i-1]\!]} (C_{k-1,j'} + d_{j',i})$
    add j in $\mathcal{J}$
    $i := j$
end for
return OPT the optimal cost and the set of selected indexes $\mathcal{J}$.

Proposition 7. *Let $E = \{x_i\}_{i \in [\![1,n]\!]}$ be a 2D PF, let $p \geqslant 2$. Algorithm 1 solves p-dispersion-MSN in $O(pn^2)$ time and $O(pn)$ memory space.*

Proof. Let $p \geqslant 3$ and let us prove the validity of Algorithm 1. Induction formula (23) uses only values $C_{i,j}$ with $j < k$. Hence, $C_{k,n}$ is at the end of each loop in k the optimal value of k-dispersion-MSN among the n points of E, and the optimal cost is given by $C_{p,n}$. The remaining operations consist in a standard backtrack algorithm to return an optimal solution. This proves the validity of Algorithm 1 to solve optimally p-dispersion-MSN.

Let us analyze the complexity. Re-indexing E following Lemma 1 has a time complexity in $O(n \log n)$. Computing the line $k = 2$ of the DP matrix has also a time complexity in $O(n)$. Computing $\max_{j \in [\![k-1, i-1]\!]} (C_{k-1,j} + d_{j,i})$ is in $O(i - k)$ and thus in $O(n)$ enumerating all the $i - k$ possibilities. It induces time complexities in $O(pn^2)$ for the construction of the DP matrix, and in $O(pn)$ for the backtracking operations. Finally, the time complexity

is given by the construction of the DP matrix, in $O(pn^2)$ time. The space complexity is in $O(pn)$, storing the DP matrix C. □

In Algorithm 1, the DP matrix C is computed line by line, with index k increasing. The computation of line $k+1$ requires only line k and $O(1)$ computations of distances. To compute only the optimal value $C_{p,n}$, it is possible to delete the line $k-1$ once the line k is completed. Such implementation has a spatial complexity in $O(n)$ with at most $2n$ elements in memory. One may have only one line in memory, with an "in-place" implementation, computing values $C_{k,m}$ for a given k with index m decreasing: in-place m'th values for $m' < m$ are still $C_{k-1,m'}$ that are needed to compute $C_{k,m}$. Algorithm 1 has a spatial complexity in $O(pn)$ because the backtracking operations use the full DP matrix. Algorithm 2 has a $O(n)$ memory space algorithm by adapting techniques that were used in [11, 21]. Algorithm 2 stores an intermediate value in the middle of the path of an optimal solution, to recover an optimal solution with a recursive divide and conquer strategy which will not be penalizing for the asymptotic time complexity. Such recursion applied to index $j = \text{argmax}_{j' \in [\![p-1, N-1]\!]}(C_{p-1,j'} + d_{j',N})$ is valid, but it would lead to a $O(p^2n^2)$ time complexity to have a space complexity on $O(n)$.

Let $p' = \lfloor \frac{p}{2} \rfloor$. We define a DP matrix H with $H_{k,m}$ for $m \in [\![1, N]\!]$ and $k \in [\![p'; p]\!]$ an index in $[\![1, m]\!]$ such that there is an optimal solution of k-dispersion-MSN among points in $[\![1, m]\!]$ such that the first p' selected points are an optimal solution of p'-dispersion-MSN among points indexed in $[\![1, H_{k,m}]\!]$. We have following induction relations:

$$\forall i \in [\![p', n]\!], \qquad H_{p',i} = i \tag{25}$$

$$\forall k > p',\ \forall i \in [\![k, n]\!], \qquad H_{k,i} = H_{k-1,\text{argmax}_{j \in [\![k-1, i-1]\!]}(C^{\text{MSN}}_{k-1,j} + d_{j,i})} \tag{26}$$

Algorithm 2. p-dispersion-MSN in a 2D PF using $O(n)$ space.

Input:
– n points of $\mathbb{R}^2$, $E = \{x_i\}_{i \in [\![1,n]\!]}$ a re-indexed 2D PF;
– an integer p with $2 \leqslant p \leqslant n$.
– $a, b \in [\![1, n]\!]$ with $a < b$
Output: optimal solution and cost of p-dispersion-MSN in $\{x_i\}_{i \in [\![a,b]\!]}$

```
DivideConquer(E, a, b, p)
    if p = 2: return d_{a,b}, {a, b}
    initialize vector C with C_i := d_{a,i} for all i ∈ [[a, b]]
    initialize vector H with H_i := i for all i ∈ [[a, b]]
    for k = 3 to p − 1 with increment k ← k + 1:
        for i = b to a + k − 2 with increment i ← i − 1:
            j := argmax_{j ∈ [[a+k−3, i−1]]}(C_j + d_{j,i})
            C_i := C_j + d_{j,i}
        if k > ⌊p/2⌋: H_i := H_j
        end for
    end for
    j := argmax_{j ∈ [[p−1, N−1]]}(C_j + d_{j,b}); OPT := C_j + d_{j,b}; h := H_j
    delete vectors C and H
    if p = 3: return d_{a,b} + d_{j,b}, {a, j, b}
    else if p = 4: return d_{a,h} + d_{h,j} + d_{j,b}, {a, h, j, b}
    else:
        OPT_1, J_1 := DivideConquer(E, a, h, ⌊p/2⌋)
        OPT_2, J_2 := DivideConquer(E, h, j, p − 1 − ⌊p/2⌋)
        return OPT, J_1 ∪ J_2 ∪ {b}.
```

Theorem 1. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a 2D PF. Max-Sum-Neighbor p-dispersion is solvable in polynomial time in the 2D PF E. The cases $p = 2, 3$ are solvable in $O(n)$ time using an $O(1)$ additional memory space. When $p > 3$, p-dispersion-MSN is solvable in $O(pn^2)$ time and $O(n)$ memory space.*

Proof. Cases $p = 2, 3$ are given by Propositions 1 and 3, so that we suppose $p \geqslant 4$ and we consider Algorithm 2, calling DivideConquer$(E, 1, n, p)$ after a $O(n \log n)$ time re-indexing using Lemma 1, which will not influence the final complexity. The validity and complexity of the cost computations arez given by Proposition 7. By induction and using (25), (26), we prove easily that H_m for $m \in [\![1, N]\!]$ is at the end of loop $k \in [\![p', p-1]\!]$ an index in $[\![1, m]\!]$ such that there is an optimal solution of k-dispersion-MSN among points in $[\![1, m]\!]$ such that the $\lfloor \frac{p}{2} \rfloor$ first selected points are an optimal solution of $\lfloor \frac{p}{2} \rfloor$-dispersion-MSN in $[\![1, H_m]\!]$. For the last iteration p, the value of $H_{p,n}$ is $h = H_j$. The validity of the backtracking operations is given by induction. The terminal cases $p = 2$ and $p = 3$ are given by Propositions 1 and 3. The terminal case with $p = 4$ is given with the extreme points, j is the third point $(3 = 4 - 1)$ computed by OPT, and the second point is h $(2 = \lfloor \frac{4}{2} \rfloor)$. By induction, the optimal solution is concatenated using that x_b is an optimal selected point, having an optimal solution of p'-dispersion-MSN calling DivideConquer(E, a, h, p') and it remains to compute a $(p-p'-1)$-dispersion-MSN between x_h and x_j. This proves the validity by induction.

Space complexity is in $O(n)$ with C and H vectors in the first cost computation, and thereafter the space usage decreases (C and H are deleted before the recursive calls). Key point is the time complexity. Let $T(n, p)$ the computation time to compute p-dispersion-MSN among n points.

Using Proposition 7, there exists β such that βpn^2 is an upper bound for the computation of OPT. Hence, We have following induction relation:

$$T(n, p) \leqslant \beta pn^2 + T(H_{p,n}, p') + T(j - H_{p,n}, p - p' - 1) \tag{27}$$

By induction, we can prove that it exists $\gamma \geqslant 2 \times \beta$ such that for all n, p, $T(n, p) \leqslant \gamma pn^2$. It is true for terminal conditions and by induction:

$$T(n, p) \leqslant \beta pn^2 + \gamma p'(H_{p,n})^2 + \gamma(p - p' - 1) \times (n - H_{p,n})^2 \leqslant \gamma pn^2. \tag{28}$$

Hence, Algorithm 2 runs in $O(pn^2)$ time using $O(n)$ space. □

Remark 4. Using Algorithm 2 instead of Algorithm 1, if we have the same asymptotic time complexity, the number of operations (and thus the CPU time) is approximatively doubled. As mentioned by [21], this should be used only if memory is missing to use Algorithm 1.

6. p-dispersion-Mm is polynomially solvable in a 2D PF

Lemma 2 implies Bellman equations for Max-min p-dispersion:

Proposition 8. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. Defining $C^{\mathrm{Mm}}_{k,i}$ as the optimal cost of Max-min k-dispersion among the points re-indexed in $[\![1, i]\!]$ for all $k \in [\![2, p]\!]$ and $i \in [\![k, n]\!]$, we have following relations:*

$$\forall i \in [\![1, n]\!], \qquad C^{\mathrm{Mm}}_{2,i} = d_{1,i} \tag{29}$$

$$\forall k \in [\![3, p]\!],\ \forall i \in [\![k, n]\!], \qquad C^{\mathrm{Mm}}_{k,i} = \max_{j\in[\![k-1,i-1]\!]} \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right). \tag{30}$$

Proof. Equation (29) is given by Proposition 1. We suppose $k \geqslant 3$ and prove (30). Let $i \in [\![k, n]\!]$. Selecting for each $j \in [\![k-1, i-1]\!]$ an optimal solution of $(k-1)$-dispersion-Mm in $\{x_l\}_{l\in[\![1,j]\!]}$, and adding point i, it defines a feasible solution for k-dispersion-Mm in $\{x_l\}_{l\in[\![1,i]\!]}$ with a cost $\min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$. This last cost is lower than the optimal k dispersion cost, thus $C^{\mathrm{Mm}}_{k,i} \geqslant \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$. Therefore, $C^{\mathrm{Mm}}_{k,i} \geqslant \max_{j\in[\![k-1,i-1]\!]} \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$.

Let $j_1 < j_2 < \cdots < j_{k-1} < j_k$ be indexes defining an optimal solution of k-dispersion-Mm, its cost is $C^{\mathrm{Mm}}_{k,i}$. Using Proposition 2, we can assume that $j_1 = 1$ and $j_k = i$. Let c be the $(k-1)$-dispersion-Mm of points indexed by $j_1 < j_2 < \cdots < j_{k-1}$. We have $c \leqslant C^{\mathrm{Mm}}_{k-1,j_{k-1}}$. If $c \geqslant d_{j_{k-1},j_k}$, the bottleneck distance is given by points indexed by j_{k-1}, j_k and we have $C^{\mathrm{Mm}}_{k,i} = d_{j_{k-1},j_k} = \min\left(C^{\mathrm{Mm}}_{k-1,j_{k-1}}, d_{j_{k-1},i}\right)$. Otherwise, $j_1, j_2, \ldots, j_{k-1}$ define an optimal solution of $(k-1)$-dispersion-Mm among points indexed in $[\![1, j_{k-1}]\!]$. On the contrary, a strictly better solution for $C^{\mathrm{Mm}}_{k,i}$ would be constructed adding the index $i = j_k$. We have thus in this case: $C^{\mathrm{Mm}}_{k,i} = \min\left(C^{\mathrm{Mm}}_{k-1,j_{k-1}}, d_{j_{k-1},i}\right)$. This is also true in the case $c \geqslant d_{j_{k-1},j_k}$, this is thus always true. It proves the reverse inequality : $C^{\mathrm{Mm}}_{k,i} \leqslant \max_{j\in[\![k-1,i-1]\!]} \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$. □

As in Algorithm 1, equations (29) and (30) allow to design a DP algorithm with a complexity in $O(pn^2)$ time and $O(pn)$ space. Following developments improve this complexity. Firstly, the time complexity is improved with a logarithmic search in Algorithm 3:

Algorithm 3. Computation of $\max_{j\in[\![k-1,i-1]\!]} \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$.

```
define a := k − 1, b := i
while b − a ⩾ 2
  Compute j = ⌊(a+b)/2⌋
  if C^Mm_{k−1,j} − d_{j,i} > 0  then  b := j
    else  a := j
end while
return max(min(C^Mm_{k−1,a}, d_{a,i}), min(C^Mm_{k−1,b}, d_{b,i}))
```

Proposition 9. *Let $k \in [\![3,p]\!]$ and $i \in [\![k,n]\!]$. Algorithm 3 computes $C^{\mathrm{Mm}}_{k,i} = \max_{j\in[\![k-1,i-1]\!]} \min\left(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i}\right)$ with a time complexity in $O(\log(i+1-k))$ once the $C^{\mathrm{Mm}}_{k-1,j}$ are computed for all $j \in [\![k-1, i-1]\!]$.*

Proof. Let $k \in [\![3,p]\!]$ and $i \in [\![k,n]\!]$. Lemma 2 ensures that the application $j \in [\![k-1,i]\!] \mapsto d_{j,i}$ is strictly decreasing. The application $j \in [\![k-1,i]\!] \mapsto C^{\mathrm{Mm}}_{k-1,j}$ is increasing: any feasible solution of $(k-1)$-dispersion-Mm among the j first points, is a feasible solution for $(k-1)$-dispersion-Mm considering the $j+1$ first points, and the optimal value $C^{\mathrm{Mm}}_{k-1,j}$ is increasing. Hence, $\varphi_{i,k} : j \in [\![k-1,i]\!] \mapsto C^{\mathrm{Mm}}_{k-1,j} - d_{j,i}$ is strictly increasing. Let $\psi_{i,k} : j \in [\![k-1,i]\!] \mapsto \min(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i})$. Note that $\varphi_{i,k}(i) = C^{\mathrm{Mm}}_{k-1,i} > 0$. Let $\alpha = \min\{j \in [\![k-1,i]\!], \varphi_{i,k}(j) \geqslant 0\}$. For $j \geqslant \alpha$, $\psi_{i,k}(j) = d_{j,i}$, and $\psi_{i,k}$ is strictly decreasing for $j \geqslant \alpha$. For $j < \alpha$, $\psi_{i,k}(j) = C^{\mathrm{Mm}}_{k-1,j}$, and ψ is increasing for $j < \alpha$. Hence, $\psi_{i,k}$ reaches a maximum for $j = \alpha$ or $j = \alpha - 1$. The computation of α, as the minimal value such that $\varphi_{i,k}(j) \geqslant 0$, can be solved with a dichotomic search presented in Algorithm 3, for a time complexity in $O(\log(i+1-k))$. □

To have a linear space complexity, one can design a recursive DP algorithm as in Algorithm 2. For Max-min p-dispersion, simple greedy algorithms in Algorithms 4 and 4' are valid as backtracking procedures:

Inputs of Algorithms 4 and 4': $E = \{x_i\}_{i\in[\![1,n]\!]}$ a re-indexed 2D PF;
$p \in [\![3;n]\!]$ and OPT, the optimal cost of Max-min p-dispersion.
Output: an optimal solution given by the selected indexes.

Proposition 10. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. Let $p \in [\![3;n]\!]$. Once the optimal cost of Max-min p-dispersion problem is computed, Algorithms 4 and 4' compute an optimal solution in $O(p \log n)$ time using $O(p)$ additional memory space. Furthermore, let $j_1 = 1, j_2, \ldots, j_{p-1}, j_p = n$ be the indexes of an optimal solution, let $1, i_2, \ldots, i_{p-1}, n$ (resp $1, i'_2, \ldots, i'_{p-1}, n$) be the indexes given by Algorithms 4 and 4'. We have:*

$$\forall k \in [\![2, p-1]\!], \qquad i_k \leqslant j_k \leqslant i'_k. \tag{31}$$

Algorithm 4. Backtracking algorithm using $O(n)$ space.

initialize $M := 1$, $m := 1$, $\mathcal{S} = \{1, n\}$.
for $k = 2$ to $p - 1$ with increment $k \leftarrow k + 1$
  $M :=$ the smallest index such that $d(x_m, x_M) \geqslant \text{OPT}$
  add M to $\mathcal{S}$
  $m := M$
end for
return $\mathcal{S}$

Algorithm 4'. Backtracking algorithm using $O(n)$ space.

initialize $M := n$, $m := n$, $\mathcal{S} = \{1, n\}$.
for $k = p - 1$ to 2 with increment $k \leftarrow k - 1$
  $m :=$ the biggest index such that $d(x_m, x_M) \geqslant OPT$
  add m to $\mathcal{S}$
  $M := m$
end for
return $\mathcal{S}$

In other words, the indexes given by Algorithms 4 and 4' are lower and upper bounds of the indexes of any optimal solution of Max-min p-dispersion considering the extreme points x_1 and x_n.

Proof. We prove the result for Algorithm 4, proof for Algorithm 4' is similar. Let $j_1 = 1, j_2, \ldots, j_{p-1}, j_p = n$ be the indexes of an optimal solution, let $i_1 = 1, i_2, \ldots, i_{p-1}, i_p = n$ be the indexes given by Algorithm 4. Firstly, we prove by induction on k that for all $k \in [\![1, p-1]\!]$, $i_k \leqslant j_k$. The case $k = 1$ is given by $j_1 = i_1 = 1$. We suppose $k > 1$ and that the induction hypothesis is true for $k-1$, *i.e.* $i_{k-1} \leqslant j_{k-1}$. The index i_k is the smallest index such that $d(x_{i_k}, x_{i_{k-1}}) \geqslant \text{OPT}$. Using Lemma 2 and $i_{k-1} \leqslant j_{k-1}$, $d(x_{j_k}, x_{j_{k-1}}) \leqslant d(x_{j_k}, x_{i_{k-1}})$. Having $i_k > j_k$ would be in contradiction with $d(x_{j_k}, x_{j_{k-1}}) \geqslant \text{OPT}$ and the definition of i_k as the smallest index such that $d(x_{i_k}, x_{i_{k-1}}) \geqslant \text{OPT}$. We have also $i_k \leqslant j_k$, which terminates the induction proof, indexes i_k are lower bounds of indexes j_k.

Let us prove that indexes $i_1 = 1, i_2, \ldots, i_{p-1}, i_p = n$ define an optimal solution. By construction $d(x_{i_k}, x_{i_{k-1}}) \geqslant \text{OPT}$ for all $k \in [\![1, p-1]\!]$, we have just to prove that $d(x_n, x_{i_{p-1}}) \geqslant \text{OPT}$. Having $i_{p-1} \leqslant j_{p-1} \leqslant j_p = n = i_p$, Lemma 2 implies $d(x_n, x_{i_{p-1}}) \geqslant d(x_n, x_{j_{p-1}})$. Optimality implies $d(x_n, x_{j_{p-1}}) \geqslant \text{OPT}$, and thus $d(x_n, x_{i_{p-1}}) \geqslant \text{OPT}$.

Let us analyze the complexity. Algorithm 4 calls at most $p-2$ times the computation of smallest index i_k such that $d(x_{i_k}, x_{i_{k-1}}) \geqslant \text{OPT}$, which can be proceeded with a dichotomic search, it runs in $O(p \log n)$ time. □

Using Proposition 10, Algorithm 5 is a valid DP algorithm for Max-min p-dispersion running in $O(n)$ memory space. Using Algorithms 4 and 4' instead of a divide-and-conquer strategy avoids to double the computation times , as noticed for p-dispersion-MSN. Theorem 2 summarizes the complexity results for Max-min p-dispersion:

Theorem 2. *Let $E = \{x_i\}_{i \in [\![1,n]\!]}$ be a 2D PF. Max-min p-dispersion is polynomially solvable to optimality in the 2D PF E. Cases $p = 2, 3$ are solvable with a complexity in $O(n)$ time using an additional memory space in $O(1)$. With $p > 3$, Algorithm 5 solves Max-min p-dispersion with a complexity in $O(pn \log n)$ time and $O(n)$ space.*

Proof. The cases $p = 2, 3$ are given by Propositions 1 and 3, so that we suppose $p \geqslant 4$ and we consider Algorithm 5. The induction formula (30) in Proposition 8 uses only values $C^{\text{Mm}}_{k-1,j}$ in Algorithm 5. At the end of each iteration of the loop in k, it holds that $C_i = C^{\text{Mm}}_{k,i}$ for all $i \in [\![k; n-1]\!]$. The optimal cost is given by a last computation with Algorithm 3 to give $C^{\text{Mm}}_{p,n}$. The validity of the backtracking procedures is proven in Proposition 10.

Algorithm 5. Max-min p-dispersion in a 2D-PF with $p \geqslant 3$.

Input: n points of a 2D PF, $E = \{x_i\}_{i\in[\![1,n]\!]}$; an integer $p \in [\![3, n]\!]$.
re-index E following the order of Lemma 1
initialize vector C with $C_i = d_{1,i}$ for $i \in [\![2; n-1]\!]$
for $k = 3$ to $p-1$ with increment $k \leftarrow k+1$:
  for $i = n-1$ to k with increment $i \leftarrow i-1$:
    $C_i := \max_{j\in[\![k-1,i-1]\!]} \min(C_j, d_{j,i})$ with Algorithm 3
  end for
end for
OPT $:= \max_{j\in[\![p,n-1]\!]} \min(C_j, d_{j,n})$ with Algorithm 3
return OPT and a solution of Algorithm 4 (or Algorithm 4')

Let us analyze the complexity of Algorithm 5. The space complexity is in $O(n)$, storing at most one line of the DP matrix C^{Mm}. Re-indexing E with Lemma 1 has a time complexity in $O(n \log n)$. Computing the line $k = 2$ of the DP matrix has a time complexity in $O(n)$. The other lines are computed in $O(n \log n)$ time with Proposition 9, for a total computation of the DP matrix in $O(pn \log n)$ time. The backtracking operations run in $O(p \log n)$ time, so that the time complexity is given in $O(pn \log n)$ time. □

7. p-DISPERSION-MSm IS POLYNOMIALLY SOLVABLE IN A 2D PF

To have a DP algorithm for Max-Sum-min p-dispersion, more adaptation is needed as the cost computation of adding a new point i depends on the choice of the next selected point $i' > i$. To design a DP algorithm, we do not store partial optimal solution of Max-Sum-min k-dispersion in a subset of points. We define $C^{\mathrm{MSm}}_{k,i,i'}$ as the best partial cost of k-dispersion-MSm in $E' = \{x_i\}_{i\in[\![1,i']\!]}$ with $i < i'$ knowing that i is the last selected point before i' and without counting a cost for point i'.

Proposition 11. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a re-indexed 2D PF. Let $p \in [\![3, n]\!]$. For all $k \in [\![3, p]\!]$, $i' \in [\![k, n]\!]$ and $i \in [\![k-1, i'-1]\!]$, Defining $C^{\mathrm{MSm}}_{k,i,i'}$ as mentioned before, we have the following induction relations:*

$$\forall i' \in [\![3, n]\!],\ \forall i \in [\![2, i'-1]\!], \qquad C^{\mathrm{MSm}}_{3,i,i'} = d_{1,i} + \min(d_{1,i}, d_{i,i'}) \tag{32}$$

$\forall k \in [\![4, p]\!], \forall i' \in [\![k, n]\!], \forall i \in [\![k-1, i'-1]\!]$,

$$C^{\mathrm{MSm}}_{k,i,i'} = \max_{j\in[\![k-2,i-1]\!]} C^{\mathrm{MSm}}_{k-1,j,i} + \min(d_{j,i}, d_{i',i}). \tag{33}$$

Proof. Using Proposition 2, $C^{\mathrm{MSm}}_{3,i,i'}$ is defined selecting $1, i, i'$, this makes the partial dispersion removing the $d_{i,i'}$ term as given in (32). Let $k \in [\![4, p]\!]$, let $i' \in [\![k, n]\!]$ and let $i \in [\![k-1, i'-1]\!]$. Selecting for each $j \in [\![k-2, i-1]\!]$ an optimal partial solution of $(k-1)$-dispersion-MSm among points indexed in $[\![1, i]\!]$ with j as last selected point before i, and adding point i, it makes a feasible solution for $(k-1)$-dispersion-MSm among points indexed in $[\![1, i]\!]$ with a partial cost $C^{\mathrm{MSm}}_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$. This last cost is lower than the optimum $C^{\mathrm{MSm}}_{k,i,i'} \geqslant C^{\mathrm{MSm}}_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$. Therefore,

$$C^{\mathrm{MSm}}_{k,i,i'} \geqslant \max_{j\in[\![k-2,i-1]\!]} C^{\mathrm{MSm}}_{k-1,j,i} + \min(d_{j,i}, d_{i',i}). \tag{34}$$

Let $j_1, \ldots, j_k$ be indexes such that $1 \leqslant j_1 < j_2 < \cdots < j_{k-1} = i < j_k = i'$ defining an optimal partial solution of k-dispersion-MSm in $[\![1, i']\!]$ with i as last selected point before i', its cost is $C^{\mathrm{MSm}}_{k,i,i'}$. Necessarily, $j_1, j_2, \ldots, j_{k-1}$ defines an optimal partial solution of $(k-1)$-dispersion-MSm among points indexed in $[\![1, i]\!]$ with j_{k-2} as last selected point before i. On the contrary, a strictly better solution for $C^{\mathrm{MSm}}_{k,i,i'}$ would be constructed adding the index i'. We have thus: $C^{\mathrm{MSm}}_{k,i,i'} = C^{\mathrm{MSm}}_{k-1,j_{k-2},i} + \min(d_{j_{k-2},i}, d_{i',i})$. Combined with (34), it proves: $C^{\mathrm{MSm}}_{k,i} = \max_{j\in[\![k-2,i-1]\!]} C^{\mathrm{MSm}}_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$. □

Bellman equations of Proposition 11 allow to solve p-dispersion-MSm in a 2D PF with a DP algorithm detailed in Algorithm 6. Once DP matrix $C^{\text{MSm}}_{k,i,i'}$ is computed, the optimal value of the complete Max-Sum-min p-dispersion is the best value $C_{p,j,n} + d_{j,n}$ for $j < n$.

Algorithm 6. Max-Sum-min p-dispersion in a 2D-PF with $p > 5$.

Input: $E = \{x_i\}_{i\in[\![1,n]\!]}$ a 2D PF of size n; an integer $p \in [\![6, n]\!]$.
re-index E following the order of Lemma 1
initialize matrix C with $C_{k,i,i'} := 0$ for all $k \in [\![2, p-1]\!]$, $(i, i') \in [\![1, n]\!]^2$
for $i' = 3$ to n
  for $i = 2$ to $i' - 1$
    $C_{3,i,i'} := d_{1,i} + \min(d_{1,i}, d_{i,i'})$
  end for
end for
for $k = 4$ to p
  for $i' = k$ to n
    for $i = k - 1$ to $i' - 1$
      $C_{k,i,i'} := \max_{j\in[\![k-2,i-1]\!]} C_{k-1,j,i} + \min(d_{j,i}, d_{i,i'})$
    end for
  end for
end for
$j := \text{argmax}_{j'\in[\![p-2,n-1]\!]}(C_{p,j',n} + d_{j',n})$
OPT $:= C_{p,j,n} + d_{j,n}$
initialize $i' := n$, $i := j$ and $\mathcal{S} := \{1, j, n\}$.
for $k = p - 1$ to 3 with increment $k \leftarrow k - 1$
  compute $j :=\text{argmax}_{j'\in[\![k-2,i-1]\!]} C_{k-1,j',i} + \min(d_{j',i}, d_{i',i})$
  add j in $\mathcal{S}$; $i' := i$; $i := j$
end for
return OPT the optimal cost and the set of selected indexes $\mathcal{S}$.

Theorem 3. *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a 2D PF. Max-Sum-min p-dispersion is polynomially solvable to optimality in the 2D PF E. Using an additional memory space in $O(1)$, cases $p = 2, 3$ are solvable in $O(n)$ time, case $p = 4$ (resp. $p = 5$) are solvable in $O(n^2)$ (resp. $O(n^3)$) time. Algorithm 6 solves the cases $p > 5$ with a complexity in $O(pn^3)$ time and $O(pn^2)$ space.*

Proof. The cases $p = 2, 3, 4, 5$ are given by Propositions 1, 3 and 4, so that we suppose $p \geqslant 6$ and we consider Algorithm 6. The proof of the validity of Algorithm 6 to compute the optimal value and an solution for Max-Sum-min p-dispersion is similar to Theorems 1 and 2. Algorithm 6 computes optimal values of $C^{\text{MSm}}_{k,i,i'}$ with k increasing requiring only $C^{\text{MSm}}_{k-1,i,i'}$ values. By induction, it proves that for all k, $C^{\text{MSm}}_{k,i,i'}$ has the optimal value at the end of the loop k. The optimal value of Max-Sum-min p-dispersion in E is then given by $\max_{j\in[\![p-1,N-1]\!]}(C_{p,j,n} + d_{j,n})$. The remaining operations define a standard backtrack algorithm. This proves the validity of Algorithm 6.

Let us analyze the complexity. The space complexity is in $O(pn^2)$, storing the DP matrix C. The time complexity is given by the construction of the matrix $C^{\text{MSm}}_{k,i,i'}$, *i.e.* $O(pn^2)$ operations running in $O(n)$ time with naive enumerations to compute $C_{k,i,i'} = \max_{j\in[\![k-2,i-1]\!]} C_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$. Algorithm 6 has thus a time complexity in $O(pn^3)$. □

A corollary is that these algorithms also apply in 1D, proving for the first time the polynomial complexity of Max-Sum-min p-dispersion in 1D:

Corollary 1 (p-dispersion-MSm is polynomial in 1D)**.** *Let $E = \{x_i\}_{i\in[\![1,n]\!]}$ be a set of n distinct real numbers, let $p \geqslant 2$. The Max-Sum-min p-dispersion problem is polynomially solvable in E. The cases $p = 2, 3$ are solvable*

with a complexity in $O(n)$ time using an additional memory space in $O(1)$. The cases $p = 4$ (resp $p = 5$) are solvable with a complexity in $O(n^2)$ (resp $O(n^3)$) time using an additional memory space in $O(1)$. The cases $p > 5$ are solvable in $O(pn^3)$ time and $O(pn^2)$ memory space.

Proof. By applying Algorithm 6 and Theorem 3 for $E' = \{(0, x_i)\}_{i\in[\![1,n]\!]} = \{(0, x_1), \ldots, (0, x_n)\}$, it would give the result with a valid algorithm. However, this degenerate case of PF is not considered by our assumptions. Therefore, we use an alternative definition of E' to conform to the assumptions of this article. Using the Euclidean distance and denoting $M = \max_{j\in[\![1,n]\!]} x_j$, we consider the affine 2D PF:

$$E' = \left\{ \left(\frac{M - x_i}{\sqrt{2}}, \frac{x_i}{\sqrt{2}} \right) \right\}_{i\in[\![1,n]\!]}. \tag{35}$$

With this definition, we still have:

$$\forall (i, j) \in [\![1, n]\!]^2, \qquad d\left(\left(\frac{M - x_i}{\sqrt{2}}, \frac{x_i}{\sqrt{2}} \right), \left(\frac{M - x_j}{\sqrt{2}}, \frac{x_j}{\sqrt{2}} \right) \right) = |x_i - x_j| \tag{36}$$

so that p-dispersion-MSm in E is the same problem than considering p-dispersion-MSm in E'. □

Algorithm 6 uses a memory in $O(pn^2)$. To compute only the optimal cost, a space complexity in $O(n^2)$ is obtained deleting elements $C_{k-1,j',i}$ when all the elements $C_{k,i,i'}$ are computed. As for Algorithm 2, it is possible to decrease this space complexity in $O(n^2)$, using recursion and alternative backtrack operations. Such algorithm is presented in Appendix A.

8. Discussions

In this section, we discuss some theoretical insights and practical applications of Theorems 1–3 and Algorithms 1, 2, 5 and 6.

8.1. Equivalent solutions and hierarchic *p*-dispersion

Proposition 10 gives tight bounds for the indexes of optimal solution of Max-min p-dispersion in a 2D PF. Many optimal solutions may exist for p-dispersion-Mm. Having an optimal solution, one can identify the bottleneck distance and rearrange other selected points without changing Max-min dispersion. The solutions of Algorithms 4 and 4' are very unbalanced, leading to the largest values for the last calculated distances. For a practical application, it is natural to wish well-balanced solutions.

In order to achieve this, a bi-objective hierarchic optimization can rank optimal solutions of p-dispersion-Mm with dispersion-MSN. Bellman equations of Propositions 6 and 8 can be extended. Indeed, we can define DP matrix pairs $(C'^{Mm}_{k,i}, C'^{\mathrm{MSN}}_{k,i})$ denoting the optimal costs with the lexicographic order, optimizing firstly k-dispersion-Mm, and then k-dispersion-MSN among points $\{x_j\}_{j\in[\![1,i]\!]}$ indexed in $[\![1, i]\!]$. We have $C'^{\mathrm{Mm}}_{k,i} = C^{\mathrm{Mm}}_{k,i}$. Such DP matrices are constructed in $O(pn^2)$, enumerating for each computation i, k possible costs with an intermediate index j, and sorting the best current value with lexicographic order. Backtracking as in Algorithm 1, the hierarchic optimization is also running in $O(pn^2)$ time and $O(pn)$ space.

Another way to have well-balanced solutions is to design a polishing heuristic starting from the solutions given by Algorithms 4 or 4'. One may use a local polishing procedure for a better balance considering 3-dispersion-Mm optimizations for consecutive points. Each 3-dispersion-Mm computation is running in $O(\log n)$ using Proposition 9 as points are already sorted. Computing the optimal consecutive Max-min 3-dispersions runs in $O(p \log n)$, so that even with $O(n)$ such iterations, the total complexity remains in $O(pn \log n)$ time and $O(n)$ space.

8.2. Implementation and parallelization issues

Time complexity in $O(pn \log n)$ or $O(pn^2)$ can be satisfactory for large scale p-dispersion computations. For a practical speed-up, Algorithms 1, 2, 5 and 6 have useful properties for efficient implementation and parallelization. The bottleneck in time complexity is the construction of the DP matrices. The backtracking algorithm is essentially sequential, but with a low complexity; this phase is not crucial for the global efficiency. The initial sorting algorithm is significant in the computation times only for Max-min p-dispersion with small values of p. Standard parallelization of sorting algorithms applies, even using General Purpose Graphical Processing Units (GPGPU) [51].

The construction of the DP matrix requires independent computations to compute $(k+1)$-dispersion costs from k-dispersion values. In Algorithm 1 (and for the lexicographic optimization), there are $O(n^2)$ independent operations in the kth loop. There are $O(n^3)$ independent operations in the loop k of Algorithm 5. After the specific improvements for Max-min p-dispersion with Algorithms 2 and 4, there are $O(n)$ independent operations to compute each value $C^{Mm}_{k,i}$ for $i \in [\![k, n]\!]$ in $O(\log i)$ time. In all cases, the parallelization is straightforward in a shared memory environment like OpenMP or in a distributed environment using Message Passign Interface (MPI). Parallel implementation requires only $p-3$ synchronizations; this is a good property for the practical efficiency. Applying LPT (Lowest Processing Times) rules from [27] for load balancing among the operations that can be computed in parallel, it is better to calculate the most time-consuming operations first, starting from the highest values of i down to the lowest.

DP Algorithms 1, 2 and 5 are cache-friendly for an efficient implementation. Indeed, it requires only k-dispersion values to compute $(k+1)$-dispersion costs. Since these k-dispersion previous values are called several times, it is crucial to access them quickly. Having the k-dispersion values in cache allows such quick access. As written in Algorithm 1, one have to cache two vectors of size n for an implementation keeping the previous and current lines of the DP matrix in cache. With in-place implementations as in Algorithms 2 and 4, one may cache only one vector of size n, which is useful when cache size becomes a limiting factor. In backtracking operations of Algorithm 1, the elements are likely no longer cached, inducing more access time. With the lower complexity of backtracking, this is not a problem.

Finally, we discuss the possibility of massive parallelization under GPGPU. Exhaustive enumerations as in p-dispersion-MSN (and the lexicographic optimization) is compatible with GPGPU parallelization. However, this is not the case for the dichotomic search in Algorithm 3. For Max-min p-dispersion, one may parallelize the $O(pn \log n)$ time version with OpenMP or the $O(pn^2)$ time under GPGPU. Finally, note that a line by line computation of the DP matrix as in Algorithm 2 is useful for GPGPU parallelization: memory in the GPU hardware may be a limiting factor.

8.3. From 2D PF to 3D PF

In this paper, we considered 2D PF generated by bi-objective optimization. We analyze here the possible extension of the results for larger dimensions, for an application to MOO problems with three or more objectives. 3D PF are generated in many real-world optimization problems, for instance maximizing robustness and stability and minimizing financial costs of maintenance planning [15]. General 2D p-dispersion problems is a sub-case of 3D PF: these are affine 3D PF. As in the proof of Corollary 1, a similar transformation applies to consider general 2D instances as 3D PFs. $\mathcal{NP}$-hard complexity for such cases with Max-min p-dispersion implies that Max-min p-dispersion is also $\mathcal{NP}$-hard for 3D PF. Unless $\mathcal{P} = \mathcal{NP}$, there is no hope to generalize DP algorithms with a polynomial complexity to PF in dimension three and more. The 1/2 approximation factor is valid in 3D (and higher dimensions) PF for Max-min and Max-Sum p-dispersion problems, as it holds for any metric space [53, 54].

Lemmas 1 and 2 are fundamental results to design DP algorithms and are highly specific for 2D PF cases, no such total order exists in 3D and larger dimensions. Generally, this explains why clustering and selection problems are polynomial in 2D PF and $\mathcal{NP}$-hard in larger dimensions. Proposition 1 eases the calculation of p-dispersion problems with the selection of extreme points, which is crucial for Propositions 2–5. An open

question is whether Proposition 1 could be extended to 3 dimensions (and higher dimensions). Extreme points are generically defined by different lexicographic optimization with permutations of the considered objectives. An extension of Proposition 1 would be to analyze if optimal solutions of p-dispersion problems in a PF should necessarily contain such extreme points. Actually, the answer is negative, as shown by the following counter-example. For $n = 5$ and with 3-dispersion problems, we consider points $(10, 0, 0)$; $(5, 5, 0)$; $(0, 10, 0)$; $(9.99, 0.01, 1)$ and $(0.01, 9.99, 1)$. There are here only two extreme points (out of the $3! = 6$ lexicographic minimizations): $(10, 0, 0)$ and $(0, 10, 0)$, whereas each 3-dispersion variant has the same and unique optimal solution $(5, 5, 0)$; $(9.99, 0.01, 1)$ and $(0.01, 9.99, 1)$.

Lastly, the possible use of 2D PF DP algorithms as heuristics is discussed for 3D PF and higher dimensions. In general, one can project any PF structure into 1D structures, for example with weighted-sum scalarization, Principal Component Analysis (PCA) or linear regression. It allows to initialize a local search approach with solutions obtained after a projection in 1D, as in [32]. The projection of a 3D PF to a representative 2D PF is difficult in general, affine 3D PF represent any planar instance without any regularity. This perspective would hold only for some specific 3D PFs.

8.4. Having continuous 2D PFs

The first hypothesis of our paper was to have a 2D PF of size n. To address complexity results for p-dispersion problems (and also for k-means, k-medoids, k-center variants) in the general and in the 2D PF case, a finite number of points shall be considered. This hypothesis can be reformulated as "let n points from a 2D PF" to define the problem. In the context of PFs, this finite hypothesis may be in contradiction with the possibility of having infinite PFs in MOO problems. In continuous PFs, p-dispersion problems make sense as continuous optimization problems. If bounded discrete MOO problems induce finite PFss, this is not the case with MO Linear Programs (MOLP) or MO MILPs [20]. For MOLPs, PFs are given as a connected subset of the frontier of a polyhedron, which is described by a finite number of extreme points of this polyhedron. In 2D, extreme points define the extremity of consecutive segments. For MO MILPs, PFs may be composed of isolated points and PFs of sub MOLPs. Continuous MOO problems may also give an analytic formula of PFs. To address p-dispersion problems in such a context, one may sample regularly the continuous parts of PFs to have a good discrete approximation of the dispersion problem. By analyzing which maximum value of n induces reasonable computation times depending on the context, this guides the choice of the granularity of such a discretization.

8.5. Application to k-medoids/k-means clustering in a 2D PF

Exact DP algorithms for k-medoids run in $O(n^3)$ time [17]. Such complexity is a bottleneck for the practical applications with large values of n. Classical heuristics for k-means/k-medoids problems may be used in such cases [9,48]. Such heuristics have no guarantee of optimality, and depend heavily on initialization strategies [9]. One may initialize local search with p-dispersion solutions to have a quick initialization procedure. Several such initialization strategies are possible. Firstly, one can initialize the k centroids using k-dispersion. Secondly, one can solve $2k + 1$-dispersion, and select the k intermediate points following order of Lemma 1. Thirdly, one can solve $3k$-dispersion, and select the $3k' + 1$ intermediate points for $k' \in [\![0; k - 1]\!]$. For these strategies, one may use optimal DP algorithms or local search iterations of 3-dispersion as in Section 8.1. Based on the preprint version of this paper, numerical results were provided for randomly generated 2D PFs with $n \leqslant 5000$. Good results and very quick computation times are obtained with such heuristics, whereas computing optimal solution with exact DP is time consuming for $n = 5000$ [32]. Having many different initialization procedures is useful for the final quality of solutions, implementing several local searches with different initialization strategies in parallel environments, as in [14]. This is also an interest of the numerical results presented in [32].

8.6. Applications to MOO meta-heuristics

In the case of population MOO meta-heuristics like EAs, selection operators may be called iteratively among the current n non-dominated points to operate cross-overs or mutations (or applying a trajectory local search)

among a restricted number of $p \ll n$ solutions [52]. Randomized selection operators may be used in that goal [52]. Deterministic strategies can also provide representative and diverse solutions, in addition to or as a replacement for random selection operators for some iterations. In such context, Selecting such p points algorithms running in $O(pn \log n)$ time are of major interest in such context, it is the case with Max-min p-dispersion, continuous p-center problems [19], and hypervolume subset selection [34]. Discrete p-center is even faster to compute in $O(n \log n)$ time [8]. Note that it is not required to have optimal solutions of such problems inside EAs. Even k-medoids heuristics may be used as in [32] as long as the calculation time of the heuristic remains fast, complexity calculations guide for such an heuristic design. Polishing procedures are of interest to have better balanced solutions as presented in Section 8.1. The fastest DP algorithms may be used to initialize local search heuristics for other problems, as in [32].

Having several types of deterministic selection operators is of interest for EAs to diversify the points where mutations or intensification are operated among consecutive iterations. On the contrary, it is a source of inefficiency to call the operators of mutations and selections always on the same points. Varying the value of p gives a first solution of this problem for deterministic operators. Selecting points in a 2D PF, many optimization measures and algorithms are available, and have different properties; another solution is to solve iteratively different selection problems. When selecting points in a PF, the rule of thumb is using hypervolume measures and variants [24,29]. HSS has useful properties to explain such popularity [29]. Clustering with k-means or k-medoids define dense zones in the PF where little intensification is needed [17]. Partial p-center variants detect outliers (isolated points) simultaneously with a rough definition of clusters [19]. In a 2D PF, isolated points should be preferred to try intensification strategies in such zones, to have better-balanced points along the 2D PF [19]. Without the partial extension, p-center problems are faster to compute, but the clustering and selection are less relevant; fast DP algorithms for p-center problems are useful to design quick heuristics for other selection or clustering optimization. HSS and p-dispersion measures find diverse points in the 2D PF, but can lead to very different solutions: HSS would avoid points that are close to local Nadir points (of knee-points) contrary to p-dispersion problems.

An open perspective is to design EAs combining randomized selection operators with different deterministic and complementary strategies.

9. Conclusion and perspectives

The properties of the four standard p dispersion problems have been examined in a 2D PF using Euclidean, Minkowski or Chebyshev distances. A novel variant, namely Max-Sum-Neighbor p-dispersion, is defined specifically for 2D PF. Cases $p = 2$ and $p = 3$ induce a complexity in $O(n)$ time. Cases $p = 4$ and $p = 5$ have respectively time complexities in $O(n^2)$ and $O(n^3)$ time. Such results are useful to select a small number of representative solutions for decision makers.

Three variants are proven solvable in polynomial time in a 2D PF, with the design of DP algorithms. Standard Max-min p-dispersion problem is solvable in a 2D PF in $O(pn \log n)$ time and using $O(n)$ memory space. Max-Sum-Neighbor p-dispersion is solvable in a 2D PF in $O(pn^2)$ time and $O(n)$ space. A lexicographic optimization considering Max-min and Max-Sum-Neighbor p-dispersion is also solvable in $O(pn^2)$ time using $O(pn)$ space. Max-Sum-min p-dispersion is solvable in a 2D PF in $O(pn^3)$ time. This last result and the DP algorithm proves also that Max-Sum-min p-dispersion is polynomially solvable in 1D, which was never studied before. Perspectives may be to improve this complexity result using specificity of 1D instances. Considering Max-Sum p-dispersion, the $\mathcal{NP}$-hardness of 2D PF and also 2D sub-cases are still open questions.

These results are not only of a theoretical interest, but raise also practical perspectives. Complexity for Max-min and Max-Sum-Neighbor p-dispersion allows a straightforward application for large 2D PF. Furthermore, DP algorithms have useful properties for an efficient implementation, including efficient parallelization in a multi or many-core environment. It allows an application inside MOO population meta-heuristics to archive partial PF at each iteration. In this context, p-dispersion DP algorithms may be used also to initialize k-medoids clustering in a 2D PF. The extension of results to higher dimensions was discussed. 3D PF cases are $\mathcal{NP}$-hard

and approximation algorithms with factor 1/2 are available. Perspectives are only to design quick heuristics for PF in such dimensions. Lastly, the results of this paper may be extended to implicit versions of the problem related to Skyline Operator and MOO applications.

Appendix A. DP with quadratic memory space for p-dispersion-MSm

Similarly to Algorithm 2, memory space of DP in Algorithm 6 can be reduced from $O(pn^2)$ to $O(n^2)$. Let $p' = \lfloor \frac{p}{2} \rfloor$. We define DP matrices H, H' with $H_{k,i,i'}$ and $H'_{k,i,i'}$ defined for $k \in [\![p'; p]\!]$, $i' \in [\![k, n]\!]$ and $i \in [\![k-1, i'-1]\!]$ such that there is an optimal solution of k-dispersion-MSm among points in $[\![1, i']\!]$ and selecting i just before i' such that the p' first selected points are an optimal solution of p'-dispersion-MSN in $[\![1, H'_{k,i,i'}]\!]$ selecting $H_{k,i,i'}$ as p'th point . Such definition induces following induction relations:

$$\forall i' \in [\![p', n]\!], \quad \forall i \in [\![p'-1, i'-1]\!] \quad H_{p',i,i'} = i \tag{A.1}$$

$$\forall i' \in [\![p', n]\!], \quad \forall i \in [\![p'-1, i'-1]\!] \quad H'_{p',i,i'} = i'. \tag{A.2}$$

Denoting for $k > p', i' \in [\![p', n]\!], i \in [\![p'-1, i'-1]\!]$:

$$j_{k,i,i'} = \text{argmax}_{j' \in [\![k-2, i-1]\!]} C_{k-1,j',i} + \min(d_{j',i}, d_{i',i}) :$$

$$\forall k > \left\lfloor \frac{p}{2} \right\rfloor, \quad \forall i' \in [\![p', n]\!], \forall i \in [\![p'-1, i'-1]\!] \qquad H_{k,i,i'} = H_{k-1,j_{k,i,i'},i} \tag{A.3}$$

Algorithm 7. p-dispersion-MSm in a 2D PF using $O(n^2)$ space.

DivideConquer$(E, prev, a, b, next, p)$
  if $p = 2$
    return $(\min(d_{prev,a}, d_{j,a}) + \min(d_{j,a}, d_{j,b}) + \min(d_{j,b}, d_{b,next}),\ \{a, b\}$
  if $p = 3$:
    $j' :=$argmax$_{j \in [\![a+1, b-1]\!]}(\min(d_{prev,a}, d_{j,a}) + \min(d_{j,a}, d_{j,b}) + \min(d_{j,b}, d_{b,next}))$
    return $\min(d_{prev,a}, d_{j',a}) + \min(d_{j',a}, d_{j',b}) + \min(d_{j',b}, d_{b,next}),\ \{a, j', b\}$
  initialize matrices H, H' with $H_{i,i'} := i$, $H'_{i,i'} := i'$ for all $i \in [\![a; b-1]\!], i' \in [\![i+1; b]\!]$
  initialize C with $C_{i,i'} := \min(d_{prev,a}, d_{a,i}) + \min(d_{a,i}, d_{i,i'})$ for $i \in [\![a; b-1]\!], i' \in [\![i+1; b]\!]$
  for $k = 3$ to $p-1$
    for $i' = b$ to $a + k$
      for $i = a + k - 1$ to $i' - 1$
        $j' :=$ argmax$_{j \in [\![1, i-1]\!]} C_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$
        $C_{k,i,i'} := C_{k-1,j,i} + \min(d_{j,i}, d_{i',i})$
        if $k > \lfloor \frac{p}{2} \rfloor$: $H_{i,i'} := H_{j,i}$; $H'_{i,i'} := H'_{j,i}$
        end for
    end for
  end for
  $j :=$ argmax$_{j' \in [\![p-1, n-1]\!]}\ C_{p-1,j',b} + \min(d_{j',b}, d_{b,next})$;
  OPT $:= C_{p,j,b} + \min(d_{j,b}, d_{b,next})$
  $h := H_{j,b}$
  $h' := H'_{j,b}$
  delete matrices C, H, H'
  if $p = 4$: **return** OPT, $\{a, h, j, b\}$
  if $p = 5$: **return** OPT, $\{a, h, h', j, b\}$
  else:
    OPT$_1, \mathcal{J}_1 :=$DivideConquer$\left(E, prev, a, h, h', \lfloor \frac{p}{2} \rfloor\right)$
    OPT$_2, \mathcal{J}_2 :=$DivideConquer$\left(E, h, h', j, b, p - 1 - \lfloor \frac{p}{2} \rfloor\right)$
    return OPT, $\mathcal{J}_1 \cup \mathcal{J}_2 \cup \{b\}$.

$$\forall k > \left\lfloor \frac{p}{2} \right\rfloor, \quad \forall i' \in [\![p', n]\!], \forall i \in [\![p'-1, i'-1]\!] \qquad H'_{k,i,i'} = H'_{k-1,j_{k,i,i'},i}. \tag{A.4}$$

Another difficulty is for the recursion that computation of costs with extreme depends on the previous and next points in the partial solution. DivideConquer$(E, prev, a, b, next, p)$ compute the optimal p-dispersion-MSm in $\{x_i\}_{i\in[\![a,b]\!]}$, knowing that the point before a is $prev$, and the point after b is $next$. Having $a = 1$ or $b = n$ disregard these values, that are set to $prev = -1$ and $next = n + 1$. To ease some reading, we take convention $d_{-1,i} = +\infty$ and $d_{i,n+1} = +\infty$. The optimal solution of p-dispersion-MSm is then obtained calling DivideConquer$(E, -1, 1, n, n + 1, p)$.

Acknowledgements. The author wishes to thank Professor Arie Tamir for his useful suggestions based on the ArXiv preprint. The author gratefully thanks the anonymous reviewers for their helpful comments that allowed to improve the paper very significantly.

References

[1] P. Agarwal, M. Sharir and E. Welzl, The discrete 2-center problem. *Discrete Comput. Geom.* **20** (1998) 287–305.
[2] S. Ağca, B. Eksioglu and J. Ghosh, Lagrangian solution of maximum dispersion problems. *Nav. Res. Logist.* **47** (2000) 97–114.
[3] A. Auger, J. Bader, D. Brockhoff and E. Zitzler, Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences, in Proceedings of GECCO 2009, ACM (2009) 563–570.
[4] C. Bazgan, F. Jamain and D. Vanderpooten, Discrete representation of the non-dominated set for multi-objective optimization problems using kernels. *Eur. J. Oper. Res.* **260** (2017) 814–827.
[5] S. Borzsony, D. Kossmann and K. Stocker, The skyline operator, in Proceedings 17th International Conference on Data Engineering, IEEE (2001) 421–430.
[6] K. Bringmann, S. Cabello and M. Emmerich, Maximum volume subset selection for anchored boxes. Preprint arXiv:1803.00849 (2018).
[7] K. Bringmann, T. Friedrich and P. Klitzke, Two-dimensional subset selection for hypervolume and epsilon-indicator, in Annual Conference on Genetic and Evolutionary Computation, ACM (2014) 589–596.
[8] S. Cabello, Faster distance-based representative skyline and k-center along Pareto Front in the plane. *J. Glob. Optim.* (2023) 1–26 https://doi.org/10.1007/s10898-023-01280-1.
[9] M. Celebi, H. Kingravi and P. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Syst. Appl.* **40** (2013) 200–210.
[10] R. Chandrasekaran and A. Daughety, Location on tree networks: p-centre and n-dispersion problems. *Math. Oper. Res.* **6** (1981) 50–57.
[11] J. Choi, S. Cabello and H.-K. Ahn, Maximizing dominance in the plane and its applications. *Algorithmica* **83** (2021) 3491–3513.
[12] A. Denstad, E. Ulsund, M. Christiansen, L. Hvattum and G. Tirado, Multi-objective optimization for a strategic ATM network redesign problem. *Ann. Oper. Res.* **296** (2019) 7–33.
[13] S. Doddi, M. Marathe, S. Ravi, D. Taylor and P. Widmayer, Approximation algorithms for clustering to minimize the sum of diameters. *Nord. J. Comput.* **7** (2000) 185–203.
[14] N. Dupin and E.-G. Talbi, Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. *Int. Trans. Oper. Res.* **27** (2020) 219–244.
[15] N. Dupin and E.-G. Talbi, Matheuristics to optimize refueling and maintenance planning of nuclear power plants. *J. Heuristics* **27** (2021) 63–105.
[16] N. Dupin, F. Nielsen and E.-G. Talbi, k-Medoids and p-median clustering are solvable in polynomial time for a 2d Pareto Front. Preprint arXiv:1806.02098 (2018).
[17] N. Dupin, F. Nielsen and E.-G. Talbi, k-Medoids clustering is solvable in polynomial time for a 2d Pareto Front, in World Congress on Global Optimization, Springer (2019) 790–799.
[18] N. Dupin, F. Nielsen and E.-G. Talbi, Clustering a 2d Pareto Front: p-center problems are solvable in polynomial time, in International Conference on Optimization and Learning, Springer (2020) 179–191.
[19] N. Dupin, F. Nielsen and E.-G. Talbi, Unified polynomial dynamic programming algorithms for p-center variants in a 2D Pareto Front. *Mathematics* **9** (2021) 453.
[20] M. Ehrgott and X. Gandibleux, Multiobjective combinatorial optimization – theory, methodology, and applications, in Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, Springer (2003) 369–444.
[21] J. Erickson, Advanced Dynamic Programming (2020). http://jeffe.cs.illinois.edu/teaching/algorithms/notes/D-faster-dynprog.pdf (accessed 28 sept 2022).
[22] E. Erkut, The discrete p-dispersion problem. *Eur. J. Oper. Res.* **46** (1990) 48–60.
[23] E. Erkut and S. Neuman, Comparison of four models for dispersing facilities. *INFOR: Inf. Syst. Oper. Res.* **29** (1991) 68–86.
[24] J. Falcón-Cardona and C. Coello, Indicator-based multi-objective evolutionary algorithms: a comprehensive survey. *ACM Comput. Surv. (CSUR)* **53** (2020) 1–35.

[25] G. Frederickson, Parametric search and locating supply centers in trees, in Workshop on Algorithms and Data Structures, Springer (1991) 299–319.
[26] M. Gibson, G. Kanade, E. Krohn, I.A. Pirwani and K. Varadarajan, On metric clustering to minimize the sum of radii. *Algorithmica* **57** (2010) 484–498.
[27] R. Graham, Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **17** (1969) 416–429.
[28] A. Grønlund, K. Larsen, A. Mathiasen, J. Nielsen, S. Schneider and M. Song, Fast exact k-means, k-medians and Bregman divergence clustering in 1D. Preprint arXiv:1701.07204 (2017).
[29] A. Guerreiro, C. Fonseca and L. Paquete, The hypervolume indicator: problems and algorithms. Preprint arXiv:2005.00515 (2020).
[30] P. Hansen and I. Moon, Dispersing facilities on a network, Cahiers du GERAD (1995).
[31] R. Hassin and A. Tamir, Improved complexity bounds for location problems on the real line. *Oper. Res. Lett.* **10** (1991) 395–402.
[32] J. Huang, Z. Chen and N. Dupin, Comparing local search initialization for k-means and k-medoids clustering in a planar Pareto Front, a computational study, in International Conference on Optimization and Learning, Springer (2021) 14–28.
[33] M. Kuby, Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems. *Geograph. Anal.* **19** (1987) 315–329.
[34] T. Kuhn, C. Fonseca, L. Paquete, S. Ruzika, M. Duarte and J. Figueira, Hypervolume subset selection in two dimensions: formulations and algorithms. *Evol. Comput.* **24** (2016) 411–425.
[35] T. Lei and R. Church, On the unified dispersion problem: efficient formulations and exact algorithms, *Eur. J. Oper. Res.* **241** (2015) 622–630.
[36] X. Lin, Y. Yuan, Q. Zhang and Y. Zhang, Selecting stars: the k most representative skyline operator, in 2007 IEEE 23rd International Conference on Data Engineering, IEEE (2007) 86–95.
[37] M. Magnani, I. Assent and M. Mortensen, Taking the big picture: representative skylines based on significance and diversity. *VLDB J.* **23** (2014) 795–815.
[38] M. Mahajan, P. Nimbhorkar and K. Varadarajan, The planar k-means problem is NP-hard. *Theor. Comput. Sci.* **442** (2012) 13–21.
[39] N. Megiddo, Linear-time algorithms for linear programming in R3 and related problems. *SIAM J. Comput.* **12** (1983) 759–776.
[40] N. Megiddo and K. Supowit, On the complexity of some common geometric location problems. *SIAM J. Comput.* **13** (1984) 182–196.
[41] N. Megiddo and A. Tamir, New results on the complexity of p-centre problems. *SIAM J. Comput.* **12** (1983) 751–758.
[42] F. Nielsen, Output-sensitive peeling of convex and maximal layers. *Inf. Process. Lett.* **59** (1996) 255–259.
[43] T. Peugeot, N. Dupin, M.-J. Sembely and C. Dubecq, MBSE, PLM, MIP and robust optimization for system of systems management, application to SCCOA French air defense program, in Complex Systems Design & Management, Springer (2017) 29–40.
[44] D. Pisinger, Upper bounds and exact algorithms for p-dispersion problems. *Comput. Oper. Res.* **33** (2006) 1380–1398, 2006.
[45] S. Ravi, D. Rosenkrantz and G. Tayi, Heuristic and special case algorithms for dispersion problems. *Oper. Res.* **42** (1994) 299–310.
[46] S. Sayın, Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Math. Prog.* **87** (2000) 543–560.
[47] D. Sayah and S. Irnich, A new compact formulation for the discrete p-dispersion problem. *Eur. J. Oper. Res.* **256** (2017) 62–67.
[48] E. Schubert and P. Rousseeuw, Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms. Preprint arXiv:1810.05691 (2018).
[49] O. Schuetze, C. Hernandez, E. Talbi, J. Sun, Y. Naranjani and F. Xiong, Archivers for the representation of the set of approximate solutions for MOPs. *J. Heuristics* **25** (2019) 71–105.
[50] D. Shier, A min-max theorem for *p*-center problems on a tree. *Transp. Sci.* **11** (1977) 243–252.
[51] E. Sintorn and U. Assarsson, Fast parallel GPU-sorting using a hybrid algorithm. *J. Parallel Distrib. Comput.* **68** (2008) 1381–1388.
[52] E. Talbi, Metaheuristics: From Design to Implementation. Vol. 74, Wiley (2009).
[53] A. Tamir, Obnoxious facility location on graphs. *SIAM J. Discrete Math.* **4** (1991) 550–567.
[54] A. Tamir, Comments on the paper: "Heuristic and special case algorithms for dispersion problems" by SS Ravi, DJ Rosenkrantz, and GK Tayi. *Oper. Res.* **46** (1998) 157–158.
[55] G. Valkanas, A.N. Papadopoulos and D. Gunopulos, Skydiver: a framework for skyline diversification, in Proceedings of the 16th International Conference on Extending Database Technology (2013) 406–417.
[56] D. Wang and Y. Kuo, A study on two geometric location problems. *Inf. Process. Lett.* **28** (1988) 281–286.

[57] E. Zio and R. Bazzo, A clustering procedure for reducing the number of representative solutions in the Pareto Front of multiobjective optimization problems. *Eur. J. Oper. Res.* **210** (2011) 624–634.