Clustering a 2d Pareto Front: p-center problems are solvable in polynomial time

Nicolas Dupin[1],Frank Nielsen[2],El-Ghazali Talbi[3]

[1] LRI, Université Paris-Sud, Université Paris-Saclay, France,
[2] Sony Computer Science Laboratories Inc, Tokyo, Japan,
[3] Univ. Lille, UMR 9189 - CRIStAL , F-59000 Lille, France
dupin@lri.fr, Frank.Nielsen@acm.org, el-ghazali.talbi@univ-lille.fr

Abstract. Having many non dominated solutions in bi-objective optimization problems, this paper aims to cluster the Pareto front using Euclidian distances. The p-center problems, both in the discrete and continuous versions, become solvable with a dynamic programming algorithm. Having N points, the complexity is proven in $O(KN \log N)$ (resp $O(KN \log^2 N)$) time and $O(N)$ memory space for the continuous (resp discrete) K-center problem for $K \geqslant 3$. 2-center problems have complexities in $O(N \log N)$. Furthermore, parallel implementations allow practical speed-up for these algorithms.
Keywords : Optimization; Clustering algorithms ; Dynamic programming ; p-center problems ; bi-objective optimization ; Pareto front

1 Introduction

Multi-objective optimization (MOO) approaches may generate large sets of non dominated solutions using Pareto dominance [23]. A Pareto Front (PF) denotes the projection of the non dominated solutions in the space of the objectives. For a presentation to decision makers, concise information on the shape of solutions are required. One can present clusters, density of points in the clusters and representative points from the clusters. Clustering and selecting points in a PF is also a crucial issue inside MOO population heuristics [2, 25]. In this paper, we consider the case of two-dimensional (2d) PF, measuring distances in $\mathbb{R}^2$ using the Euclidian distance. The p-center problems, both in the discrete and continuous versions, define the cluster costs, covering the 2d PF with p identical balls while minimizing the radius of the balls to use.

The p-center problems are NP-complete in general and also for the Euclidian cases in $\mathbb{R}^2$ [13, 20, 19]. This paper proves that p-center problems in a 2d PF are solvable in polynomial time with a Dynamic Programming (DP) algorithm, and discusses properties of the DP algorithm for an efficient implementation. In Section 2, we describe the p-center problems in a 2d PF with unified notation. In Section 3, we discuss related state-of-the-art elements. In Section 4, intermediate results are presented, with a specific optimality property. In Section 5, the DP algorithm is presented with the complexity proofs and discussions on a parallel implementation. In Section 6, our contributions are summarized, discussing new perspectives. To ease the readability, some proofs are gathered in Appendix A.

2 Problem statement and notation

Without loss of generality, we consider a set $E = \{x_1, \ldots, x_N\}$ of N elements of $\mathbb{R}^2$, that is a Pareto Front obtained by the minimization of two objectives. As noticed in [9], this is characterized by the following property:

$$\forall 1 \geqslant i \neq j \geqslant N, \quad x_i \ \mathcal{I} \ x_j \tag{1}$$

where the relations $\mathcal{I}, \prec$ for all $y = (y^1, y^2), z = (z^1, z^2) \in \mathbb{R}^2$ are defined with:

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

$$y \preccurlyeq z \Longleftrightarrow y \prec z \ \text{ or } \ y = z \tag{3}$$

$$y \ \mathcal{I} \ z \Longleftrightarrow y \prec z \ \text{ or } \ z \prec y \tag{4}$$

We consider the Euclidian norm, defining for all $y = (y^1, y^2), z = (z^1, z^2) \in \mathbb{R}^2$:

$$d(y, z) = \|y - z\| = \sqrt{(y^1 - z^1)^2 + (y^2 - z^2)^2} \tag{5}$$

We define $\mathit{\Pi}_K(E)$, as the set of all the possible partitions of E in K subsets:

$$\mathit{\Pi}_K(E) = \left\{ P \subset \mathcal{P}(E) \;\middle|\; \forall p, p' \in P, \ p \cap p' = \emptyset, \ \bigcup_{p \in P} p = E \text{ and } |P| = K \right\} \tag{6}$$

K-center problems are combinatorial optimization problems indexed by $\mathit{\Pi}_K(E)$:

$$\min_{\pi \in \mathit{\Pi}_K(E)} \max_{P \in \pi} f(P) \tag{7}$$

The function f measures for each subset of E a dissimilarity among the points in the subset. The K-center problems cover the PF with K identical disks while minimizing the radius of the disks to use. Considering the discrete K-center problem, the center of the disks are points of the 2d PF:

$$\forall P \subset E, \quad f^{\mathcal{D}}_{ctr}(P) = \min_{y \in P} \max_{x \in P} \|x - y\| \tag{8}$$

The continuous K-center problem minimizes the radius of covering disks without any constraint for the center of the covering disks:

$$\forall P \subset E, \quad f^{\mathcal{C}}_{ctr}(P) = \min_{y \in \mathbb{R}^2} \max_{x \in P} \|x - y\| \tag{9}$$

We unify notations with $\gamma \in \{0, 1\}$, $\gamma = 0$ (resp 1) indicates that the continuous (resp discrete) p-center problem is used. The continuous and discrete K-center problems in a 2d PF are denoted K-γ-CP2dPF. f_γ denotes the clustering measure among $f^{\mathcal{C}}_{ctr}, f^{\mathcal{D}}_{ctr}$.

3 Related works

3.1 Solving p-center problems and complexity results

P-center problems are NP-hard [13, 19]. The discrete p-center problem in $\mathbb{R}^2$ with a Euclidian distance is also NP-hard [20]. To solve discrete p-center problems, exact algorithms based on ILP formulations were proposed [6, 11]. Exponential exact algorithms were also provided for the continuous p-center problem [7]. An $N^{O(\sqrt{p})}$-time algorithm solves the continuous Euclidean p-center problem in the plane [14]. Specific cases of p-center are solvable in polynomial time. The continuous 1-center, i.e. the minimum covering ball problem, has a linear complexity in $\mathbb{R}^2$ [18]. The discrete 1-center is solved in $O(N \log N)$ time using Voronoi diagrams [3]. The continuous and planar 2-center is solved in randomized expected $O(N \log^2 N)$ time [22]. The discrete and planar 2-center is solvable in time $O(N^{4/3} \log^5 N)$ [1]. The continuous and planar K-center on a line is solvable in $O(NK \log N)$ time and $O(N)$ space [15].

3.2 Clustering/selecting points in Pareto frontiers

Selecting representative points in PF has been studied for exact methods or meta-heuristics in MOO [21]. Clustering the PF is useful for population meta-heuristics to design operators like cross-over in evolutionary algorithms [23, 25]. Maximizing the quality of discrete representations of PF was studied in the Hypervolume Subset Selection (HSS) problem [2, 21]. The HSS problem is NP-hard in dimension 3 [4]. The 2d case is solvable in polynomial time by DP algorithm in $O(KN^2)$ time and $O(KN)$ space [2]. This time complexity was improved in $O(KN + N \log N)$ by [5] and in $O(K.(N-K) + N \log N)$ by [16]. Similar results exist when clustering a 2d PF. 2d p-median and k-medoids problems are NP hard [19]. The 2d PF cases are solvable in $O(N^3)$ time with DP algorithms [10, 9]. K-means is also NP-hard for 2d problems [17]. Under conjecture, the 2d PF case would be solvable in $O(N^3)$ time with a DP algorithm [8]. We note that an affine 2d PF is a line in $\mathbb{R}^2$, which is equivalent to 1 dimensional (1d) cases. 1d k-means is polynomially solvable with a DP algorithm in $O(KN^2)$ time and $O(KN)$ space [24], improved in $O(KN)$ time and $O(N)$ space [12]. Continuous K-center in a affine 2d PF has a complexity in $O(NK \log N)$ time and $O(N)$ space, specific case of continuous and planar K-center on a line [15].

4 Intermediate results

4.1 Indexation and distances in a 2d PF

Relations $\prec$ and $\preccurlyeq$ induce a new indexation of E with monotony properties. Lemma 1 extends properties of $\leqslant$ and $<$ in $\mathbb{R}$.

Lemma 1. *$\preccurlyeq$ is an order relation, and $\prec$ is a transitive relation:*

$$\forall x, y, z \in \mathbb{R}^2, \;\; x \prec y \;\; \textit{and} \;\; y \prec z \Longrightarrow x \prec z \tag{10}$$

Proposition 1 (Total order) *Points (x_i) can be indexed such that:*

$$\forall (i_1, i_2) \in [\![1; N]\!]^2, i_1 < i_2 \Longrightarrow x_{i_1} \prec x_{i_2} \tag{11}$$
$$\forall (i_1, i_2) \in [\![1; N]\!]^2, i_1 \leqslant i_2 \Longrightarrow x_{i_1} \preccurlyeq x_{i_2} \tag{12}$$
$$\forall (i_1, i_2, i_3) \in [\![1; N]\!]^3, i_1 \leqslant i_2 < i_3 \Longrightarrow d(x_{i_1}, x_{i_2}) < d(x_{i_1}, x_{i_3}) \tag{13}$$
$$\forall (i_1, i_2, i_3) \in [\![1; N]\!]^3, i_1 < i_2 \leqslant i_3 \Longrightarrow d(x_{i_2}, x_{i_3}) < d(x_{i_1}, x_{i_3}) \tag{14}$$

The complexity of the sorting re-indexation is in $O(N \log N)$

Proof : We index E such that the first coordinate is increasing, this sort has a complexity in $O(N \log N)$. Let's prove it implies (11) and (13), (12) is implied by (11)and proving (14) is similar with (13). Let $(i_1, i_2) \in [\![1; N]\!]^2$, with $i_1 < i_2$. We have $x^1_{i_1} < x^1_{i_2}$ with the new indexation. $x_{i_1} \mathcal{I} x_{i_2}$ implies $x^2_{i_1} > x^2_{i_2}$ and $x_{i_1} \prec x_{i_2}$, which proves (11). Let $i_1 < i_2 < i_3$. We note $x_{i_1} = (x^1_{i_1}, x^2_{i_1})$, $x_{i_2} = (x^1_{i_2}, x^2_{i_2})$ and $x_{i_3} = (x^1_{i_3}, x^2_{i_3})$. (11) implies $x^1_{i_1} < x^1_{i_2} < x^1_{i_3}$ and $x^2_{i_1} > x^2_{i_2} > x^2_{i_3}$.
Hence, $(x^1_{i_1} - x^1_{i_2})^2 < (x^1_{i_1} - x^1_{i_3})^2$ and $(x^2_{i_1} - x^2_{i_2})^2 < (x^2_{i_1} - x^2_{i_3})^2$.
$d(x_{i_1}, x_{i_2})^2 = (x^1_{i_1} - x^1_{i_2})^2 + (x^2_{i_1} - x^2_{i_2})^2 < d(x_{i_1}, x_{i_3})^2$, which proves (13). □

4.2 1-center in a 2d PF

Proposition 2 proves that 1-center problems, i.e. computing the costs $f_\gamma(E)$, has a linear complexity. It uses the Lemma 2 proven in Appendix A.

Lemma 2. *Let $P \subset E$, $P \neq \emptyset$. Let $i \leqslant i'$ such that $x_i, x_{i'} \in P$ and for all $j \in [\![i, i']\!]$, $x_i \preccurlyeq x_j \preccurlyeq x_{i'}$.*

$$f^{\mathcal{C}}_{ctr}(P) = \frac{1}{2} \|x_i - x_{i'}\| \tag{15}$$
$$f^{\mathcal{D}}_{ctr}(P) = \min_{j \in [\![i, i']\!], x_j \in P} \max \left(\|x_j - x_i\|, \|x_j - x_{i'}\| \right) \tag{16}$$

Proposition 2 *Let $\gamma \in \{0, 1\}$. 1-γ-CP2dPF has a complexity in $O(N)$ time using an additional memory space in $O(1)$.*

Proof: Using equations (15) or (16), $f_\gamma(E)$ is computed at most in $O(N)$ time once having computed the extreme elements for the order relation $\prec$. Computing these extreme points is also in $O(N)$ time, with one traversal of E. □

4.3 Optimality of interval clustering

Proposition 3 gives a common optimality property: interval clustering is optimal.

Lemma 3. *Let $\gamma \in \{0, 1\}$. Let $P \subset P' \subset E$. We have $f_\gamma(P) \leqslant f_\gamma(P')$.*

Proof: Let i, j (resp i', j') the minimal and maximal indexes of points of P (resp P'). Using Proposition 1 and Lemma 2, $f^{\mathcal{C}}_{ctr}(P) \leqslant f^{\mathcal{C}}_{ctr}(P')$ is easy to prove and using: $f^{\mathcal{D}}_{ctr}(P) = \min_{k \in [\![i,j]\!], x_k \in P} \max \left(\|x_k - x_i\|, \|x_j - x_k\| \right)$, we have:
$f^{\mathcal{D}}_{ctr}(P) \leqslant \min_{k \in [\![i',j']\!], x_k \in P'} \max \left(\|x_k - x_i\|, \|x_j - x_k\| \right)$
$f^{\mathcal{D}}_{ctr}(P) \leqslant \min_{k \in [\![i',j']\!], x_k \in P'} \max \left(\|x_k - x_{i'}\|, \|x_{j'} - x_k\| \right) = f^{\mathcal{D}}_{ctr}(P')$ □

Proposition 3 *Let $\gamma \in \{0,1\}$, let $K \in \mathbb{N}^*$, let $E = (x_i)$ a 2d PF, indexed following Proposition 1. There exists optimal solutions of K-γ-CP2dPF using only clusters on the shape $\mathcal{C}_{i,i'} = \{x_j\}_{j\in[\![i,i']\!]} = \{x \in E \mid \exists j \in [\![i,i']\!],\ x = x_j\}$.*

Proof: We prove the result by induction on $K \in \mathbb{N}^*$. For $K = 1$, the optimal solution is $E = \{x_j\}_{j\in[\![1,N]\!]}$. Let us suppose $K > 1$ and the Induction Hypothesis (IH): Proposition 3 is true for $(K-1)$-γ-CP2dPF. Let $\pi \in \Pi_K(E)$ an optimal solution of K-γ-CP2dPF, let OPT the optimal cost. We denote $\pi = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$, $\mathcal{C}_K$ being the cluster of x_N. For all $k \in [\![1,K]\!]$, $f_\gamma(\mathcal{C}_k) \leqslant OPT$. Let i the minimal index such that $x_i \in \mathcal{C}_K$. We consider the subsets $\mathcal{C}'_K = \{x_j\}_{j\in[\![i,N]\!]}$ and $\mathcal{C}'_k = \mathcal{C}_k \cap \{x_j\}_{j\in[\![1,i-1]\!]}$ for all $k \in [\![1,K-1]\!]$. $\{\mathcal{C}'_1, \dots, \mathcal{C}'_{K-1}\}$ is a partition of $E' = \{x_j\}_{j\in[\![1,i-1]\!]}$, and $\{\mathcal{C}'_1, \dots, \mathcal{C}'_K\}$ is a partition of E. For all $k \in [\![1,K-1]\!]$, $\mathcal{C}'_k \subset \mathcal{C}_k$ so that $f_\gamma(\mathcal{C}'_k) \leqslant f_\gamma(\mathcal{C}_k) \leqslant OPT$ (Lemma 3). $\mathcal{C}'_1, \dots, \mathcal{C}'_K$ is a partition of E, and $\max_{k\in[\![1,K]\!]} f(\mathcal{C}'_k) \leqslant OPT$. Hence, $\mathcal{C}'_1, \dots, \mathcal{C}'_{K-1}$ is an optimal solution of $(K$-1)-γ-CP2dPF applied to E'. Let OPT' the optimal cost, we have $OPT' \leqslant \max_{k\in[\![1,K-1]\!]} f_\gamma(\mathcal{C}'_k) \leqslant OPT$. Applying IH for $(K$-1)-γ-CP2dPF to E', we have $\mathcal{C}''_1, \dots, \mathcal{C}''_{K-1}$ an optimal solution of $(K$-1)-γ-CP2dPF in E' on the shape $\mathcal{C}_{i,i'} = \{x_j\}_{j\in[\![i,i']\!]} = \{x \in E' \mid \exists j \in [\![i,i']\!],\ x = x_j\}$. For all $k \in [\![1,K-1]\!]$, $f_\gamma(\mathcal{C}''_k) \leqslant OPT' \leqslant OPT$. $\mathcal{C}''_1, \dots, \mathcal{C}''_{K-1}, \mathcal{C}'_K$ is an optimal solution of K-γ-CP2dPF in E using only clusters $\mathcal{C}_{i,i'}$. □

Algorithm 1: Computation of $f^{\mathcal{D}}_{ctr}(\mathcal{C}_{i,i'})$

input: indexes $i < i'$

define $idInf = i$, $vInf = \|x_i - x_{i'}\|$, $idSup = i'$, $vSup = \|x_i - x_{i'}\|$
while $idSup - idInf \geqslant 2$
Compute $idMid = \lfloor \frac{idSup+idInf}{2} \rfloor$, $temp = f_{i,i'}(idMid)$, $temp2 = f_{i,i'}(idMid+1)$
if $temp < temp2$ **:** set $idSup = idMid, vSup = temp$
else if $temp > temp2$ **:** set $idInf = 1 + idMid, vInf = temp2$
else return $temp = temp2$
end while
return $\min(vInf, vSup)$

4.4 Computation of cluster costs

This section computes efficiently the costs of clusters $\mathcal{C}_{i,i'}$ used in Proposition 3. Once the PF E is indexed using Proposition 1, (15) computes a cluster cost $f^{\mathcal{C}}_{ctr}(\mathcal{C}_{i,i'})$ in $O(1)$. Using (16), cluster costs $f^{\mathcal{D}}_{ctr}(\mathcal{C}_{i,i'})$ can be computed in $O(i'-i)$ time for all $i < i'$. Lemma 4, proven in Appendix A, and Proposition 4 allow to compute $f^{\mathcal{D}}_{ctr}(\mathcal{C}_{i,i'})$ in $O(\log(i'-i))$ time once E is indexed using Proposition 1.

Lemma 4. *Let (i,i') with $i < i'$. $f_{i,i'} : j \in [\![i,i']\!] \longmapsto \max\left(\|x_j - x_i\|, \|x_j - x_{i'}\|\right)$ is strictly decreasing in a $[\![i,l]\!]$, and then is strictly increasing for $j \in [\![l+1,i']\!]$*

Proposition 4 , *Let $E = \{x_1, \ldots, x_N\}$ be N points of $\mathbb{R}^2$, such that for all $i \neq j$, $x_i \prec x_j$. Computing cost $f^{\mathcal{D}}_{ctr}(\mathcal{C}_{i,i'})$ for any cluster $\mathcal{C}_{i,i'}$ has a complexity in $O(\log(i'-i))$ time using Algorithm 1.*

Proof: Let $i < i'$. Algorithm 1 uses Lemma 4 to have a loop invariant: the existence of a minimal solution of $f_{i,i'}(j^*)$ with $idInf \leqslant j^* \leqslant idSup$. Algorithm 1 is a dichotomic search, finding the center and computing the cost of $\mathcal{C}_{i,i'}$ using at most $\log(i'-i)$ operations in $O(1)$. □

Algorithm 2: K-center clustering in a 2dPF, general DP algorithm

Input: N points of $\mathbb{R}^2$, $E = \{x_1, \ldots, x_N\}$ a 2d PF, $\gamma \in \{0, 1\}$, $K \in \mathbb{N} - \{0, 1\}$.

```
initialize matrix C^(γ) with C^(γ)_{k,i} = 0 for all i ∈ [[1; N]], k ∈ [[1; K − 1]]
sort E following the order of Proposition 1
compute C_{i,1} = f_γ(C_{1,i}) for all i ∈ [[1; N]]
for k = 2 to K − 1:
   for i = 2 to N − 1:
      set C^(γ)_{k,i} = min_{j∈[[2,i]]} max(C^(γ)_{k−1,j−1}, f_γ(C_{j,i}))
   end for
end for
set OPT = min_{j∈[[2,N]]} max(C^(γ)_{K−1,j−1}, f_γ(C_{j,N}))
set i = j = argmin_{j∈[[2,N]]} max(C^(γ)_{K−1,j−1}, f_γ(C_{j,N}))
initialize P = {[[j; N]]}, a set of sub-intervals of [[1; N]].
for k = K to 2 with increment k ← k − 1 :
   find j ∈ [[1, i]] such that max(C^(γ)_{k−1,j−1}, f_γ(C_{j,i})) is minimal
   add [[j, i]] in P
   i = j − 1
end for
return OPT the optimal cost and the partition P ∪ [[1, i]]
```

5 DP algorithm and complexity results

5.1 General DP algorithm

Proposition 3 implies a common DP algorithm for p-center problems. Defining $C^{(\gamma)}_{k,i}$ as the optimal cost of k-γ-CP2dPF clustering with k cluster among points $[\![1, i]\!]$ for all $i \in [\![1, N]\!]$ and $k \in [\![1, K]\!]$. The case $k = 1$ is given by:

$$\forall i \in [\![1, N]\!], \quad C^{(\gamma)}_{1,i} = f_\gamma(\mathcal{C}_{1,i}) \tag{17}$$

We have following induction relation, with the convention $C^{(\gamma)}_{k,0} = 0$ for all $k \geqslant 0$:

$$\forall i \in [\![1, N]\!], \forall k \in [\![2, K]\!], \quad C^{(\gamma)}_{k,i} = \min_{j \in [\![1,i]\!]} \max(C^{(\gamma)}_{k-1,j-1}, f_\gamma(\mathcal{C}_{j,i})) \tag{18}$$

Algorithm 2 uses these relations to compute the optimal values of $C_{k,i}^{(\gamma)}$. $C_{K,N}^{(\gamma)}$ is the optimal solution of K-γ-CP2dPF, a backtracking algorithm allows to compute the optimal partitions.

5.2 Computing the lines of the DP matrix

In Algorithm 2, the main issue for the time complexity is to compute efficiently $C_{k,i}^{(\gamma)} = \min_{j\in[\![2,i]\!]} \max(C_{k-1,j-1}^{(\gamma)}, f_\gamma(\mathcal{C}_{j,i}))$. Lemma 5 and Proposition 5 allows to compute each line of $C_{k,i}^{(\gamma)}$ with a complexity in $O(N \log N)$ time.

Lemma 5. *Let $\gamma \in \{0,1\}, i \in [\![4,N]\!]$, $k \in [\![2,K]\!]$. The application $g_{i,k} : j \in [\![2,i]\!] \longmapsto \max(C_{k-1,j-1}^{(\gamma)}, f_\gamma(\mathcal{C}_{j,i}))$. $g_{i,k}$ is firstly decreasing and then is increasing.*

Proof : Having $g_{i,k}(3) < g_{i,k}(2)$ and $g_{i,k}(N-1) < g_{i,k}(N)$, the result is given by the monotone properties: $j \in [\![1,i]\!] \longmapsto f_\gamma(\mathcal{C}_{j,i})$ is decreasing with Lemma 3, $j \in [\![1,N]\!] \longmapsto C_{k,j}^{(\gamma)}$ is increasing for all k, as proven below for $k \geqslant 2$, the case $k = 1$ is implied by the Lemma 3. Let $k \in [\![2,K]\!]$ and $j \in [\![2,N]\!]$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ an optimal solution of k-γ-CP2dPF among points $(x_l)_{l\in[\![1,j]\!]}$, its cost is $C_{k,j}^{(\gamma)}$. We index clusters such that $x_j \in \mathcal{C}_k$. For all $k' \in [\![1,k]\!]$, $f(\mathcal{C}_{k'}) \leqslant C_{k,j}^{(\gamma)}$. $\{\mathcal{C}'_1, \ldots, \mathcal{C}'_k\} = \{\mathcal{C}_1, \ldots, \mathcal{C}_{k-1}, \mathcal{C}_k - x_k\}$ is a partition of $(x_l)_{l\in[\![1,j-1]\!]}$, so that $C_{k,j-1}^{(\gamma)} \leqslant \max_{k'\in[\![1,k]\!]} f_\gamma(\mathcal{C}'_{k'})$. With Lemma 3, $f_\gamma(\mathcal{C}'_{k'}) = f_\gamma(\mathcal{C}_k - x_k) \leqslant f_\gamma(\mathcal{C}_k)$. Hence, $C_{k,j-1}^{(\gamma)} \leqslant \max_{k'\in[\![1,k]\!]} f_\gamma(\mathcal{C}'_{k'}) \leqslant C_{k,j}^{(\gamma)}$. □

Algorithm 3: Dichotomic computation of $\min_{j\in[\![2,i]\!]} \max(C_{k-1,j-1}^{(\gamma)}, f_\gamma(\mathcal{C}_{j,i}))$

input: indexes $i \in [\![2,N]\!]$, $k \in [\![2,K]\!]$, $\gamma \in \{0,1\}$, $v_j = C_{j,k-1}$ for $j \in [\![1,i-1]\!]$.

```
define idInf = 2, vInf = f_γ(C_{2,i}),
define idSup = i, vSup = v_{i-1},
while idSup − idInf > 2 :
    Compute idMid = ⌊(i+i')/2⌋, temp = g_{i,k}(idMid), temp2 = g_{i,k}(idMid + 1)
    if temp < temp2  : idSup = idMid, vSup = temp
        else  : idInf = 1 + idMid, vInf = temp2
end while
return min(vInf, vSup)
```

Proposition 5 (Line computation) *Let $\gamma \in \{0,1\}, i \in [\![2,N]\!], k \in [\![2,K]\!]$. Having computed the values $C_{k-1;j}^{(\gamma)}$, $C_{k,i}^{(\gamma)} = \min_{j\in[\![2,i]\!]} \max(C_{k-1,j-1}^{(\gamma)} f_\gamma(\mathcal{C}_{j,i}))$ is computed calling $O(\log i)$ cost computations $f_\gamma(\mathcal{C}_{j,i})$, for a complexity in $O(\log^\gamma i)$ time. Once the line of the DP matrix $C_{k-1,j}^{(\gamma)}$ is computed for all $j \in [\![1,N]\!]$, the line $C_{k,j}^{(\gamma)}$ can be computed in $O(N \log^{1+\gamma} N)$ time and $O(N)$ space.*

Proof: Similarly to Algorithm 1, Algorithm 3 is a dichotomic search based on Lemma 5. It calls $O(\log i)$ cost computations $f_\gamma(\mathcal{C}_{j,i})$. □

5.3 Linear memory consumption

Storing the whole matrix $C_{k,n}^{(\gamma)}$ in Algorithm 2, it uses at least a memory space in $O(KN)$. Actually, the DP matrix can be computed line by line, with k increasing. The computation of line $k+1$ requires the line k and cost computations requiring $O(1)$ additional memory space. In the DP matrix, deleting the line $k-1$ once the line k is completed allows to have $2N$ elements in the memory. It allows to compute the optimal value $C_{K,N}^{(\gamma)}$ using $O(N)$ memory space.

The backtracking operations, as written in Algorithm 2, require the whole matrix $C_{k,n}^{(\gamma)}$. Algorithm 4 provides an alternative backtrack with a complexity in $O(N)$ memory space and $O(KN \log N)$ time, as proven in Proposition 6.

Lemma 6. *Let $K \in \mathbb{N}, K \geqslant 2$. The indexes given by Algorithm 4 are lower bounds of the indexes of any optimal solution of K-γ-CP2dPF:*
Denoting $[\![1, i_1]\!], [\![i_1+1, i_2]\!], \ldots, [\![i_{K-1}+1, N]\!]$ the indexes given by Algorithm 4, and $[\![1, i'_1]\!], [\![i'_1+1, i'_2]\!], \ldots, [\![i'_{K-1}+1, N]\!]$ the indexes of an optimal solution of K-γ-CP2dPF, we have for all $k \in [\![1, K-1]\!], i_k \leqslant i'_k$.

Proof : The proof uses a decreasing induction on k. The initialisation case $k = K-1$ is given by the first step of Algorithm 4, and that $j \in [\![1, N]\!] \longmapsto f_\gamma(\mathcal{C}_{j,N})$ is decreasing with Lemma 3. Having for a given k, $i'_k \leqslant i_k$, $i_{k-1} \leqslant i'_{k-1}$ is implied by Proposition 1 and $d(z_{i_k}, z_{i_{k-1}-1}) > OPT$. □

Algorithm 4: Backtracking algorithm using $O(N)$ memory space

Input: - N points of $\mathbb{R}^2$, $\gamma \in \{0, 1\}$, $K \in \mathbb{N} - \{0, 1\}$;
- $E = \{x_1, \ldots, x_N\}$ a 2d PF indexed with Proposition 1;
- OPT, the optimal cost of K-γ-CP2dPF.

initialize $maxId = N$, $minId = N$, $\mathcal{P} = \varnothing$, a set of sub-intervals of $[\![1; N]\!]$.
for $k = K$ to 2 with increment $k \leftarrow k-1$
set $minId = maxId$
while $f_\gamma(\mathcal{C}_{minId-1,maxId})) \leqslant OPT$ **do** $minId = minId - 1$ **end while**
add $[\![minId, maxId]\!]$ in $\mathcal{P}$
set $maxId = minId - 1$
end for
return $[\![1, maxId]\!] \cup \mathcal{P}$

Proposition 6 *Knowing the optimal cost of K-γ-CP2dPF, Algorithm 4 computes an optimal partition in $O(N \log N)$ time using $O(1)$ additional space.*

Proof: Let OPT, the optimal cost of K-γ-CP2dPF. Let $[\![1, i_1]\!], [\![i_1+1, i_2]\!], \ldots, [\![i_{K-1}+1, N]\!]$ the indexes given by Algorithm 4, By construction, all the clusters $\mathcal{C}_{i_k+1,i_{k+1}}$ for all $k > 1$ verify $f_\gamma(\mathcal{C}) \leqslant OPT$. We have to prove that $f_\gamma(\mathcal{C}_{1,i_1}) \leqslant OPT$ to ensure that Algorithm 4 returns an optimal solution. Let an optimal solution, let $[\![1, i'_1]\!], [\![i'_1+1, i'_2]\!], \ldots, [\![i'_{K-1}+1, N]\!]$ the indexes defining this solution. Lemma 6

implies that $i_1 \leqslant i'_1$, and Lemma 3 implies $f_\gamma(\mathcal{C}_{1,i_1}) \leqslant f_\gamma(\mathcal{C}_{1,i'_1}) \leqslant OPT$. Analyzing the complexity, Algorithm 4 calls at most $(K+N) \leqslant 2N$ times the function f_γ, the complexity is in $O(N \log^\gamma N)$ time. □

Remark Actually, a dichotomic search can compute the largest cluster with an extremity given and a bounded cost, with a complexity in $O(K \log^{1+\gamma} N)$. To avoid the case $K = O(N)$ and $\gamma = 1$, Algorithm 4 is designed in $O(N \log N)$ time without distinguishing the cases, which is sufficient for the complexity results.

5.4 Complexity results

Theorem 1. *Let $\gamma \in \{0, 1\}$. K-γ-CP2dPF is solvable in polynomial time. 1-γ-CP2dPF is solvable in $O(N)$ time with an additional memory space in $O(1)$. 2-γ-CP2dPF is solvable in $O(N \log N)$ time and $O(N)$ space. For $K \geqslant 2$, K-γ-CP2dPF is solvable in $O(KN \log^{1+\gamma} N)$ time and $O(N)$ space.*

Proof: The induction formula (18) uses only values $C^{(\gamma)}_{j,i}$ with $j < k$ in Algorithm 3. $C^{(\gamma)}_{k,N}$ is at the end of each loop in k the optimal value of the k-center clustering among the N points of E. Proposition 6 ensures that Algorithm 4 gives an optimal partition. Let us analyze the complexity. We suppose $K \geqslant 2$, the case $K = 1$ is given by Proposition 2. The space complexity is in $O(N)$ using section 5.3. Indexing E with Proposition 1 has a time complexity in $O(N \log N)$. Computing the first line of the DP matrix costs has also a time complexity at most in $O(N \log N)$. With Proposition 5, the construction of the lines of the DP matrix $C^{(\gamma)}_{k,i}$ for $k \in [\![2, K-1]\!]$ requires $N \times (K-2)$ computations of $\min_{j \in [\![1,i]\!]} C^{(\gamma)}_{k-1,j-1} + f_\gamma(\mathcal{C}_{j,i})$, which are in $O(\log^{1+\gamma} N)$ time, the complexity of this phase is in $O((K-2)N \log^{1+\gamma} N)$. With Proposition 6, backtracking operations are also in $O(N \log N)$ time. Finally, the time complexity is in $O(N \log N + (K-2)N \log^{1+\gamma} N)$ for K-γ-CP2dPF. The case $K = 2$ induces a complexity in $O(N \log N)$ time, whereas cases $K \geqslant 3$ imply a time complexity in $O(KN \log^{1+\gamma} N)$. □

5.5 Towards a parallel implementation

The time complexity in $O(NK \log^{1+\gamma} N)$ is already efficient for large scale computations of k-γ-CP2dPF with a sequential implementation. The DP algorithm has also good properties for a parallel implementation. Computing the DP matrix line by line with Algorithm 3 requires $N-1$ independent computations to compute each line from the previous one. A parallel implementation of Algorithm 3 is straightforward using OpenMP or Message Passing Interface (MPI). The initial sort and the computation of the first line of the DP matrix, running both in $O(N \log N)$ time, can also be parallelized with OpenMP or MPI. It may have an influence on the total computation time for small values of K when the time for the sorting initialization is comparable to the time to run Algorithm 3. Algorithm 4 is sequential, but with a low complexity, so that a parallelization of this phase is not crucial.

6 Conclusion and perspectives

This paper examined properties of the p-center problems in the special case of a discrete set of non-dominated points in a 2d Euclidian space. A common optimal property is proven, allowing a resolution with a polynomial DP algorithm. A complexity is proven in $O(KN \log N)$ time for the continuous p-center and in $O(KN \log^2 N)$ time for the discrete p-center , with a space complexity in $O(N)$ for both cases. Discrete 2-center is also proven solvable in $O(N \log N)$ time. The DP algorithms can also be parallelized to speed up the computational time.

These results offer promising perspectives embedded in MOO meta-heuristics, as in [2, 25]. P-center problems are solved quickly for large PF, quicker computations are also possible with heuristics based on Algorithm 2 computations. Section 5.3 showed that many optimal solutions may exist. Choosing among the optimal solutions relevant solutions regarding other clustering measure is a perspective related to the MOO application. Designing heuristics for 3d PF is another perspective, p-center in a 3d PF being NP hard problems. Projections, as in the Johnson Lindenstrauss theorem, are a perspective to extend Algorithm 2 to greater dimensions.

References

1. P. Agarwal, M. Sharir, and E. Welzl. The discrete 2-center problem. *Discrete & Computational Geometry*, 20(3):287–305, 1998.
2. 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*, pages 563–570. ACM, 2009.
3. P. Brass, C. Knauer, H. Na, C. Shin, and A. Vigneron. Computing k-centers on a line. *arXiv preprint arXiv:0902.3282*, 2009.
4. K. Bringmann, S. Cabello, and M. Emmerich. Maximum volume subset selection for anchored boxes. *arXiv preprint arXiv:1803.00849*, 2018.
5. K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Annual Conference on Genetic and Evolutionary Computation*, pages 589–596. ACM, 2014.
6. H. Calik and B. Tansel. Double bound method for solving the p-center location problem. *Computers & operations research*, 40(12):2991–2999, 2013.
7. Z. Drezner. The p-centre problem - heuristic and optimal algorithms. *Journal of the Operational Research Society*, 35(8):741–748, 1984.
8. N. Dupin, F. Nielsen, and E. Talbi. Dynamic programming heuristic for k-means clustering among a 2-dimensional pareto frontier. *7th International Conference on Metaheuristics and Nature Inspired Computing*, pages 1–8, 2018.
9. N. Dupin, F. Nielsen, and E. Talbi. k-medoids clustering is solvable in polynomial time for a 2d Pareto front. In *World Congress on Global Optimization*, pages 790–799. Springer, 2019.
10. N. Dupin and E. Talbi. Clustering in a 2-dimensional Pareto Front: p-median and p-center are solvable in polynomial time. *arXiv preprint arXiv:1806.02098*, 2018.
11. S. Elloumi, M. Labbé, and Y. Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94, 2004.

12. A. Grønlund et al. Fast exact k-means, k-medians and Bregman divergence clustering in 1d. *arXiv preprint arXiv:1701.07204*, 2017.
13. W. Hsu and G. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.
14. R. Hwang, R. Lee, and R. Chang. The slab dividing approach to solve the Euclidean P-Center problem. *Algorithmica*, 9(1):1–22, 1993.
15. A. Karmakar et al. Some variations on constrained minimum enclosing circle problem. *Journal of Combinatorial Optimization*, 25(2):176–190, 2013.
16. T. Kuhn et al. Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary Computation*, 24(3):411–425, 2016.
17. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.
18. N. Megiddo. Linear-time algorithms for linear programming in R3 and related problems. *SIAM journal on computing*, 12(4):759–776, 1983.
19. N. Megiddo and K. Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984.
20. N. Megiddo and A. Tamir. New results on the complexity of p-centre problems. *SIAM Journal on Computing*, 12(4):751–758, 1983.
21. S. Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Math Prog*, 87(3):543–560, 2000.
22. M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete & Computational Geometry*, 18(2):125–134, 1997.
23. E. Talbi. *Metaheuristics: from design to implementation*, volume 74. Wiley, 2009.
24. H. Wang and M. Song. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal*, 3(2):29, 2011.
25. E. Zio and R. Bazzo. A clustering procedure for reducing the number of representative solutions in the pareto front of multiobjective optimization problems. *European Journal of Operational Research*, 210(3):624–634, 2011.

Appendix A: Proof of the Lemmas 2 and 4

Proof of Lemma 2: $\forall k \in [\![i,i']\!],\ \|x_j - x_k\| \leqslant \max(\|x_j - x_i\|, \|x_j - x_i\|)$, using Proposition 1.Then:
$f^{\mathcal{D}}_{ctr}(P) = \min_{j\in[\![i,i']\!],x_j\in P} \max\left(\max(\|x_j - x_i\|, \|x_j - x_i\|), \max_{k\in[\![i,i']\!]} \|x_j - x_k\|\right)$
$f^{\mathcal{D}}_{ctr}(P) = \min_{j\in[\![i,i']\!],x_j\in P} \max(\|x_j - x_i\|, \|x_j - x_{i'}\|)$. It proves (16). We prove now a stronger result than (15): the application $x \in \mathbb{R}^2 \longmapsto \max_{p\in P} \|x - p\| \in \mathbb{R}$ as a unique minimum reached for $x = \frac{x_i+x_j}{2}$:

$$\forall x \in \mathbb{R}^2 - \left\{\frac{x_i + x_j}{2}\right\},\ \max_{p\in P} \|x - p\| > \frac{1}{2}\|x_i - x_j\| = \max_{p\in P} \left\|\frac{x_i + x_j}{2} - p\right\| \quad (19)$$

We prove (19) analytically, denoting $\text{diam}(P) = \frac{1}{2}\|x_i - x_j\|$. We firstly use the coordinates defining as origin the point $O = \frac{x_i+x_j}{2}$ and that $x_i = (-\frac{1}{2}\text{diam}(P), 0)$ and $x_j = (\frac{1}{2}\text{diam}(P), 0)$. Let x a point in $\mathbb{R}^2$ and (x_1, x_2) its coordinates. We minimize $\max_{p\in P} d(x, x_p)$ distinguishing the cases:
if $x_1 > 0$, $d(x, x_i)^2 = (x_1 + \frac{1}{2}\text{diam}(P))^2 + x_2^2 \geqslant (x_1 + \frac{1}{2}\text{diam}(P))^2 > (\frac{1}{2}\text{diam}(P))^2$
if $x_1 < 0$, $d(x, x_j)^2 = (x_1 - \frac{1}{2}\text{diam}(P))^2 + x_2^2 \geqslant (x_1 - \frac{1}{2}\text{diam}(P))^2 > (\frac{1}{2}\text{diam}(P))^2$
if $x_1 = 0$ and $x_2 \neq 0$, $d(x, x_i)^2 = (\frac{1}{2}\text{diam}(P))^2 + x_2^2 > (\frac{1}{2}\text{diam}(P))^2$
In these three sub-cases, $\max_{p\in P} d(x, x_p) \geqslant d(x, x_i) > \frac{1}{2}\text{diam}(P)$. The three cases allow to reach any point of $\mathbb{R}^2$ except $x_0 = \frac{x_i+x_j}{2}$. To prove the last equality, we use the coordinates such that $x_i = (-\frac{1}{2}\text{diam}(P); 0)$ and $x_j = (\frac{1}{2}\text{diam}(P); 0)$. The origin x_0 has coordinates $(\frac{1}{2\sqrt{2}}\text{diam}(P), \frac{1}{2\sqrt{2}}\text{diam}(P))$. Let $x = (x^1, x^2) \in P$, such that $x \neq x_i, x_j$. Thus $x_i \prec x \prec x_j$. The Pareto dominance induces $0 \leqslant x_1, x_2 \leqslant \frac{1}{\sqrt{2}}\text{diam}(P)$. $d(x, x_0)^2 = (x_1 - \frac{1}{2\sqrt{2}}\text{diam}(P))^2 + (x_2 - \frac{1}{2\sqrt{2}}\text{diam}(P))^2$
$d(x, x_0)^2 \leqslant (\frac{1}{2\sqrt{2}}\text{diam}(P))^2 + (\frac{1}{2\sqrt{2}}\text{diam}(P))^2 = 2\frac{1}{8}\text{diam}(P))^2$
$d(x, x_0) \leqslant \frac{1}{2}\text{diam}(P)$, which proves (19) as $d(x_0, x_i) = d(x_0, x_j) = \frac{1}{2}\text{diam}(P)$. □

Proof of Lemma 4: Let $i < i'$. We define $g_{i,i',j}, h_{i,i',j}$ with:
$g_{i,i'} : j \in [\![i,i']\!] \longmapsto \|x_j - x_i\|$ and $h_{i,i'} : j \in [\![i,i']\!] \longmapsto \|x_j - x_{i'}\|$
Using Proposition 1, g is strictly decreasing and h is strictly increasing.
Let $A = \{j \in [\![i,i']\!] | \forall m \in [\![i,j]\!] g_{i,i'}(m) < h_{i,i'}(m)\}$.$g_{i,i'}(i) = 0$ and $h_{i,i'}(i) = \|x_{i'} - x_i\| > 0$ so that $i \in A$, $A \neq \emptyset$. We note $l = \max A$. $h_{i,i'}(i') = 0$ and $g_{i,i'}(i') = \|x_{i'} - x_i\| > 0$ so that $i' \notin A$ and $l < i'$. Let $j \in [\![i, l-1]\!]$. $g_{i,i'}(j) < g_{i,i'}(j+1)$ and $h_{i,i',j}(j+1) < h_{i,i'}(j)$. $f_{i,i'}(j+1) = \max(g_{i,i'}(j+1), h_{i,i'}(j+1)) = h_{i,i}(j+1)$ and $f_{i,i'}(j) = \max(g_{i,i'}(j), h_{i,i'}(j)) = h_{i,i}(j)$ as $j, j+1 \in A$.Hence, $f_{i,i'}(j+1) = h_{i,i'}(j+1) < h_{i,i'}(j) = f_{i,i'}(j)$. It proves that $f_{i,i'}$ is strictly decreasing in $[\![i,l]\!]$. $l+1 \notin A$ and $g_{i,i'}(l+1) > h_{i,i'}(l+1)$ to be coherent with $l = \max A$. Let $j \in [\![l+1, i'-1]\!]$. $j+1 > j \geqslant l+1$ so $g_{i,i'}(j+1) > g_{i,i'}(j) \geqslant g_{i,i'}(l+1) > h_{i,i'}(l+1) \geqslant h_{i,i'}(j) > h_{i,i'}(j+1)$. It implies $f_{i,i'}(j+1) = g_{i,i'}(j+1)$ and $f_{i,i'}(j) = g_{i,i'}(j)$ and $f_{i,i'}(j+1) < f_{i,i'}(j)$. $g_{i,i'}(j) < g_{i,i'}(j+1)$ and $h_{i,i',j}(j+1) < h_{i,i'}(j)$. $f_{i,i'}(j+1) = \max(g_{i,i'}(j+1), h_{i,i'}(j+1)) = g_{i,i}(j+1)$ and $f_{i,i'}(j) = \max(g_{i,i'}(j), h_{i,i'}(j)) = h_{i,i}(j)$ as $j, j+1 \in A$. Hence, $f_{i,i'}(j+1) = h_{i,i'}(j+1) > h_{i,i'}(j) = f_{i,i'}(j)$, $f_{i,i'}$ is strictly increasing in $[\![l+1, i']\!]$. □