k-medoids clustering is solvable in polynomial time for a 2d Pareto front

Nicolas Dupin[1], Frank Nielsen[2], and 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 - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
dupin@lri.fr, Frank.Nielsen@acm.org, el-ghazali.talbi@univ-lille.fr

Abstract. The k-medoids problem is a discrete sum-of-square clustering problem, which is known to be more robust to outliers than k-means clustering. As an optimization problem, k-medoids is NP-hard. This paper examines k-medoids clustering in the case of a two-dimensional Pareto front, as generated by bi-objective optimization approaches. A characterization of optimal clusters is provided in this case. This allows to solve k-medoids to optimality in polynomial time using a dynamic programming algorithm. More precisely, having N points to cluster, the complexity of the algorithm is proven in $O(N^3)$ time and $O(N^2)$ memory space. This algorithm can also be used to minimize conjointly the number of clusters and the dissimilarity of clusters. This bi-objective extension is also solvable to optimality in $O(N^3)$ time and $O(N^2)$ memory space, which is useful to choose the appropriate number of clusters for the real-life applications. Parallelization issues are also discussed, to speed-up the algorithm in practice.

Keywords: Bi-objective optimization · clustering algorithms · k-medoids · Euclidean sum-of-squares clustering · Pareto Front · Dynamic programming · Bi-objective clustering.

1 Introduction

This paper is motivated by real-life applications of bi-objective optimization. Some optimization problems can be driven by more than one objective function, with some conflicts among objectives. For example, one may minimize financial costs, while maximizing the robustness to uncertainties [4, 16]. In such cases, higher levels of robustness are likely to induce financial over-costs. Pareto dominance, preferring a solution from another if it is better for all the objectives, is a weak dominance rule. With conflicting objectives, several non-dominated solutions can be generated, these *efficient* solutions are the best compromises. A *Pareto front* is the projection in the objective space of the non-dominated solutions [7].

Bi-objective optimization approaches may generate large Pareto fronts, for a trade-off evaluation by a decision maker. The problem is here to select K good compromise solutions from $N \gg K$ non dominated solutions while maximizing the representativity of these K solutions. This problem can be seen as an application of clustering algorithms, partitioning the N elements into K subsets with a maximal similarity, and giving a representative element of the optimal clusters. Selecting best compromise solutions for human decision makers, one deals with small values $K < 10$. We note that partial Pareto fronts are used inside population meta-heuristics [21]. Clustering a Pareto front is also useful in this context to archive representative solutions of the partial Pareto front [23]. For such applications, the values of K are larger than the previous ones.

k-means clustering is one of the most famous unsupervised learning problem, and is widely studied in the literature since the seminal algorithm provided by Lloyd in [13]. The k-medoids problem, the discrete variant of the k-means problem, fits better with our application to maximize the dissimilarity around a representative solution [11]. If k-medoids clustering is more combinatorial than k-means clustering, it is known to be more robust on noises and outliers [10]. Both k-medoids and k-means problem are NP hard in the general and the planar case, we refer to [1] for k-means and [9] for k-medoids.

Lloyd's algorithm can be extended to solve heuristically k-medoids problems. PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications) and CLARANS (Clustering Large Applications based upon RANdomized Search) are such heuristics for k-medoids clustering [19]. Hybrid and genetic algorithms were also investigated in [20]. Previous heuristics converge only to local minima, without any guarantee to reach a global minimum.

This paper proves that the special case of k-medoids clustering in a two dimensional (2-d) Pareto front is solvable to optimality in polynomial time, using a dynamic programming (DP) algorithm. We note similarities with other works. k-center and k-median problems are also solvable in polynomial time thanks to a DP algorithm in [5]. k-means clustering in one dimension, which can be projected as an affine Pareto front in dimension 2, is also polynomial with DP [8]. A DP heuristic applies k-means in 2-d Pareto fronts [6]. Lastly, there are similarities to maximize the quality of discrete representations of Pareto sets with the hypervolume measure, giving rise to the Hypervolume Subset Selection (HSS). HSS is solvable in polynomial time for 2-d Pareto fronts thanks to a DP algorithm presented in [2].

In section 2, we define formally the problem and fix the notation. In section 3, intermediate results and a characterization of optimal clusters are presented. In section 4, it is described how to compute efficiently the costs of the previous clusters. In section 5, a DP algorithm is presented with a proven polynomial complexity. In section 6, it is discussed how to choose the value K using properties of the DP resolution. In section 7, our contributions are summarized, discussing also future directions of research.

2 Problem statement and notation

A 2-d discrete Pareto front can be defined as a set $E = \{x_1, \dots, x_N\}$ of N elements of $\mathbb{R}^2$, such that for all $i \neq j$, $x_i \ \mathcal{I} \ x_j$ defining the binary relations $\mathcal{I}, \prec$ for all $y = (y^1, y^2), z = (z^1, z^2) \in \mathbb{R}^2$ with:

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

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

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

We note that the convention leading to the definitions of $\mathcal{I}, \prec$ considered the minimization of two objectives, which is not a loss of generality. Such a set E can be extracted from any subset of $\mathbb{R}^2$ using an output-sensitive algorithm [14], or generated by bi-objective optimization approaches, exact methods [7] and also meta-heuristics [21]. We consider in this paper the Euclidian distance :

$$d(y,z) = \|y - z\| = \sqrt{(y^1 - z^1)^2 + (y^2 - z^2)^2}, \ \forall y = (y^1, y^2), (z^1, z^2) \in \mathbb{R}^2 \tag{4}$$

$\varPi_K(E)$ denotes the set of the possible partitions of E in K subsets:

$$\varPi_K(E) = \left\{ P \subset \mathcal{P}(E) \ \middle| \ \forall p, p' \in P, \ \ p \cap p' = \emptyset \, , \ \bigcup_{p \in P} p = E \text{ and } |P| = K \right\} \tag{5}$$

k-medoids clustering is a combinatorial optimization problem indexed by $\varPi_K(E)$. It minimizes the sum for all the K clusters of the dissimilarity measure $f_{mdd}(P) = \min_{c \in P} \sum_{x \in P} \|x - c\|^2$, i.e. f_{mdd} minimizes the sum of the squared distances from one chosen point of P, the *medoid*, to the other points of P :

$$\min_{\pi \in \varPi_K(E)} \sum_{P \in \pi} f_{mdd}(P) \tag{6}$$

3 Optimal property of interval clustering

In this section, optimal clusters are characterized for k-medoids clustering in a 2d Pareto Front. Intermediate lemmas with elementary proofs are first mentioned. Lemmas 1,2 and 3 are proven in [5].

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

$$\forall x, y, z \in \mathbb{R}^2, \ \ x \prec y \ \text{ and } \ y \prec z \Longrightarrow x \prec z \tag{7}$$

Lemma 2 (Total order). *Points (x_i) can be indexed such that:*

$$\forall (i_1, i_2) \in [\![1; N]\!]^2, i_1 < i_2 \Longrightarrow x_{i_1} \prec x_{i_2} \tag{8}$$

$$\forall (i_1, i_2) \in [\![1; N]\!]^2, i_1 \leqslant i_2 \Longrightarrow x_{i_1} \preccurlyeq x_{i_2} \tag{9}$$

This property is stronger than the property that $\preccurlyeq$ induces a total order in E. Furthermore, the complexity of the sorting reindexation is in $O(N. \log N)$

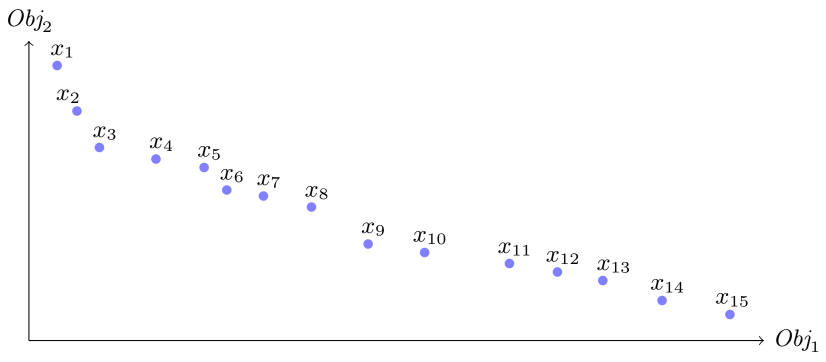

Fig. 1. Illustration of a 2-dimensional Pareto front with 15 points, minimizing two objectives and indexing the points with the Lemma 2

Lemma 3. *Let $x_1, x_2, x_3 \in \mathbb{R}^2$.*

$$x_1 \prec x_2 \prec x_3 \Longrightarrow d(x_1, x_2) < d(x_1, x_3) \tag{10}$$

Lemma 4. *Let $\mathcal{C}_1, \dots, \mathcal{C}_K$ an optimal partition with k-medoids clustering, we denote with $c_1, \dots, c_K$ the medoids. Let $x_n \in E$, we denote with $\mathcal{C}_j$ the cluster where it is assigned in the optimal clustering. Then, we have:*

$$\forall k \in [\![1, K]\!], \quad d(x_n, c_j) \leqslant d(x_n, c_k) \tag{11}$$

Proof. Ad absurdum, an elementary proof constructs easily a better partition if an equation (11) is not fulfilled. This result is well known, Lloyd's algorithm applied to k-medoids is also a hill-climbing heuristic. Hence, local minima fulfill (11), which is a more general property. □

Proposition 1 (Optimal interval clustering) *We suppose that points (x_i) are sorted following Lemma 2. The optimal solutions of the minimization problem (6) use only clusters $\mathcal{C}_{i,i'} = \{x_j\}_{j \in [\![i,i']\!]} = \{x \in E \mid \exists j \in [\![i, i']\!],\, x = x_j\}$.*

Proof. We prove the result by induction on $K \in \mathbb{N}$. For $K = 1$, the optimal cluster is $E = \{x_j\}_{j \in [\![1,N]\!]}$. Suppose $K > 1$ and the Induction Hypothesis (IH) that Proposition 1 is true for $(K-1)$-medoids clustering. We suppose having an optimal clustering partition, We denote with $\mathcal{C}$ the cluster of x_N, and x_c the medoid of $\mathcal{C}$. Let $A = \{i \in [\![1, N]\!] \mid \forall k \in [\![i, N]\!], x_k \in \mathcal{C}\}$. A is a subset of $\mathbb{N}$, non empty as $N \in A$, it has a minimum. Let $j = \min\{i \in [\![1, N]\!] | \forall k \in [\![i, N]\!], x_k \in \mathcal{C}\}$. If $j = 1$, $E = \mathcal{C} = \{x_j\}_{j \in [\![1,N]\!]}$ and the result is proven. We suppose now $j > 1$. $j - 1 \notin A$ like $j - 1 \in A$ is in a contradiction with $j = \min A$. We denote with $\mathcal{C}' \neq \mathcal{C}$ the cluster of $j - 1$, and $x_{c'}$ the center of $\mathcal{C}'$. Necessarily $c' < j$.
We prove by contradiction that $j - 1 < c$, supposing $c < j$. It implies $c < j - 1$ as $x_{j-1} \notin \mathcal{C}$, and $x_c \prec x_j$ with Lemma 3. Lemma 4 implies $d(x_{c'}, x_{j-1}) \leqslant d(x_c, x_{j-1})$ applied to x_{j-1} and $\mathcal{C}'$. With Lemma 3, this is possible only if $c < c'$.

We would have thus with Lemma 2 $x_c \prec x_{c'} \preccurlyeq x_{j-1} \prec x_j$. Applying Lemma 3, $d(x_{c'}, x_j) < d(x_c, x_j)$ is a contradiction with Lemma 4 applied to x_j and $\mathcal{C}$. It proves $j - 1 < c$ and Lemma 2 ensures that $x_{c'} \preccurlyeq x_{j-1} \prec x_j \preccurlyeq x_c$.
We prove now that for all $j' \leqslant j - 1$, $x_{j'} \notin \mathcal{C}$. Let $j' < j - 1$.
If $j' \leqslant c' < c$, Lemma 3 with $x_{j'} \preccurlyeq x_{c'} \prec x_c$ implies directly $d(x_{c'}, x_{j'}) < d(x_c, x_{j'})$ and Lemma 4 implies $x_{j'} \in \mathcal{C}$.
If $j' > c'$, we have $x_{c'} \prec x_{j'} \prec x_{j-1} \prec x_c$. $d(x_{c'}, x_{j-1}) \leqslant d(x_c, x_{j-1})$ with Lemma 4 applied to x_{j-1} and $\mathcal{C}'$. Lemma 3 implies $d(x_{c'}, x_{j'}) < d(x_{c'}, x_{j-1})$ and $d(x_c, x_{j-1}) < d(x_c, x_{j'})$. $d(x_{c'}, x_{j'}) < d(x_{c'}, x_{j-1}) < d(x_c, x_{j-1}) < d(x_c, x_{j'})$. Hence, $d(x_{c'}, x_{j'}) < d(x_c, x_{j'})$ and $x_{j'} \notin \mathcal{C}$ with Lemma 4.
This induces that for all $j' < j$, $x_{j'} \notin \mathcal{C}$. On one hand, it implies that $\mathcal{C} = \{x_l\}_{l \in [\![j,N]\!]}$. On the other hand, the other clusters are optimal for $E' = E - \mathcal{C}$ with $(K-1)$-medoids clustering. Applying IH to the $K-1$ clustering to E' proves that the optimal clusters are on the shape $\mathcal{C}_{i,i'} = \{x_j\}_{j \in [\![i,i']\!]}$. $\square$

4 Computing the costs of interval clustering

We define $c_{i,i'}$ as the cost of cluster $\mathcal{C}_{i,i'}$ for the k-medoids clustering. This section aims to compute efficiently the costs $c_{i,i'}$ for all $i < i'$. By definition:

$$\forall i < i', \quad c_{i,i'} = f_{mdd}(\mathcal{C}_{i,i'}) = \min_{j \in [\![i,i']\!]} \sum_{k \in [\![i,i']\!]} \|x_j - x_k\|^2 \tag{12}$$

The naive computation of $c_{i,i'}$ has a time complexity in $O((i'-i)^2)$. Computing $c_{i,i'}$ independently for all $i < i'$, this induces a complexity in $O(N^4)$ time and in $O(N^2)$ memory space. To improve the complexity, Algorithm 1 computes for all $i < c < i'$ the values $d_{i,c,i'}$, defined as the costs of k-medoids for cluster $\mathcal{C}_{i,i'}$ with center c, using the induction relations (14) to compute each element $d_{i,c,i'}$ in $O(1)$. The matrix $c_{i,i'}$ is then computed from $d_{i,c,i'}$ using (15):

$$\forall i \leqslant c \leqslant i', \qquad d_{i,c,i'} = \sum_{k=i}^{i'} \|x_k - x_c\|^2 \tag{13}$$

$$\forall i \leqslant c \leqslant i' < N, \quad d_{i,c,i'+1} = d_{i,c,i'} + \|x_{i'+1} - x_c\|^2 \tag{14}$$

$$\forall i \leqslant i', \qquad c_{i,i'} = \min_{l \in [\![i,i']\!]} d_{i,l,i'} \tag{15}$$

Proposition 2 *Using Algorithm 1, computing the matrix of costs $c_{i,i'}$ for all $i < i'$ has a complexity in $O(N^3)$ time and in $O(N^2)$ memory space.*

Proof. The induction formula (14) uses only values $d_{i,k,i+l'}$ with $l' < l$. In Algorithm 1, it is easy to show by induction that $d_{i,k,i+l}$, and also $c_{i,i+l}$, has its final value for all $l \in [\![1,N]\!]$ at the end of the for loops from $k = i$ to $i + l$.
Let us analyze the complexity. The space complexity is defined by the sizes of matrices $c_{i,i'}$ and $d_{i,c,i'}$. A priori, the space complexity is in $O(N^3)$, given

Algorithm 1: Computation of matrix $c_{i,i'}$ for the k-medoids problem

```
  define matrix $c$ with $c_{i,i'} = 0$ for all $(i,i') \in [\![1;N]\!]^2$ with $i \leqslant i'$
  define matrix $d$ with $d_{i,l,i'} = 0$ for all $(i,l,i') \in [\![1;N]\!]^3$ with $i \leqslant l \leqslant i'$
  for $l = 1$ to $N$ //consider subset of cardinal l
    for $i = 1$ to $N - l$
      $d_{i,i+l,i+l} = \sum_{k=i}^{i+l} \|x_k - x_{i+l}\|^2$
      for $k = i$ to $i + l - 1$
        $d_{i,k,i+l} = d_{i,k,i+l-1} + \|x_{i+l} - x_k\|^2$
      end for
      Compute $c_{i,i+l} = \min_{k \in [\![i,i+l]\!]} d_{i,k,i+l}$
    end for
  end for
return matrix $c_{i,i'}$
```

by $d_{i,c,i'}$. Actually, for a given $i, i+l$, the computations of $c_{i,i+l}$ in the inner loop requires only in memory $d_{i,c,i+l}$ and $d_{i,c,i+l-1}$. Values $d_{j,c,j+m}$ for all j and $m < l-1$ can be deleted. The order of computations increasing l in the first loop allows to consider only few variables $d_{j,c,j+m}$, in the order of $2 \times l \leqslant 2N$ to compute $c_{i,i+l}$. Hence, the space complexity is in $O(N^2)$ memory space.
Let α the time to compute $d_{i,k,i+l} = d_{i,k,i+l-1} + \|x_{i+l} - x_k\|^2$. Defining β as the time to compute an operation like $\min(d_{i,k,i'}, d_{i,k+1,i'})$ and to store the result, the time to compute $c_{i,i+l} = \min_{k \in [\![i,i+l]\!]} d_{i,k,i+l}$ is a $\beta.l$.

$$T_N = \sum_{l=1}^{N} \sum_{i=1}^{N-l} \left(\beta.l + \sum_{k=i}^{i+l} \alpha \right) = \sum_{l=1}^{N} \sum_{i=1}^{N-l} \left(\beta.l + (l+1)\alpha \right) = O(N^3) \;\square$$

To speed-up the computation of Algorithm 1, a parallel implementation is possible. Once value $l = i' - i$ is chosen, the second loop considers independent computations using results with a lower values l. This loop can be parallelized using OpenMP in a sharing-memory environment, synchronizing only when l is increased. The third loop can also be parallelized, a common parallelization with the second loop induces a better load balancing. A parallel implementation can also be designed in a distributed environment with many processors using Message Passing Interface (MPI, [15]). Partitioning $[\![1,N]\!]$ in different subsets $[\![i,i']\!]$ allows each processor to compute locally the values $c_{j,j'}$ for $i \leqslant j \leqslant j' \leqslant i'$. With neighbor processes having the neighbor sub-intervals, communications are easy to merge sub-problems. In these parallel schemes, load balancing issues are crucial using appropriate decomposition strategies [18].

5 Dynamic Programming algorithm

Let $c_{i,i'}$ for $i < i'$ the elementary costs computed with the Algorithm 1. We define $C_{i,k}$ as the optimal cost of the k-medoids clustering with k cluster among points $[\![1,i]\!]$ for all $i \in [\![1,N]\!]$ and $k \in [\![1,K]\!]$, we have following induction relation:

$$\forall i \in [\![1, N]\!], \; \forall k \in [\![2, K]\!], \quad C_{i,k} = \min_{j \in [\![1,i]\!]} C_{j-1,k-1} + c_{j,i} \tag{16}$$

with the convention $C_{0,k} = 0$ for all $k \geqslant 0$. The case $k = 1$ is directly given by:

$$\forall i \in [\![1, N]\!], \quad C_{i,1} = c_{1,i} \tag{17}$$

These relations allow to compute the optimal values of $C_{i,k}$ by dynamic programming in the Algorithm 2. $C_{N,K}$ is the optimal solution of the k-medoids problem, backtracking on the matrix $(C_{i,k})_{i,k}$ computes the optimal partitioning clusters.

Algorithm 2: k-medoids clustering in a 2d-Pareto Front

```
    initialize matrix $c$ with $c_{i,j} = 0$ for all $(i,j) \in [\![1; N]\!]^2$
    initialize matrix $C$ with $C_{i,k} = 0$ for all $i \in [\![0; N]\!], k \in [\![1; K]\!]$
    initialize $\mathcal{P}$ =nil, a set of sub-intervals of $[\![1; N]\!]$.
    sort $E$ following the order of Lemma 2
    compute $c_{i,j}$ for all $(i,j) \in [\![1; N]\!]^2$ with Algorithm 1

    for $i = 1$ to $N$    //Construction of the matrix $C$
      set $C_{i,1} = c_{1,i}$
      for $k = 2$ to $K$
        set $C_{i,k} = \min_{j \in [\![1,i]\!]} C_{j-1,k-1} + c_{j,i}$
      end for
    end for

    $i = N$    //Backtrack phase
    for $k = K$ to 1 with increment $k \leftarrow k - 1$
      find $j \in [\![1, i]\!]$ such that $C_{i,k} = C_{j-1,k-1} + c_{j,i}$
      add $[\![j, i]\!]$ in $\mathcal{P}$
      $i = j - 1$
    end for
return $C_{N,K}$ the optimal cost and the partition $\mathcal{P}$ giving the cost $C_{N,K}$
```

Theorem 1. *Let $E = \{x_1, \dots, x_N\}$ a subset of N points of $\mathbb{R}^2$, such that for all $i \neq j$, $x_i \mathcal{I} x_j$. Clustering E with k-medoids is solvable to optimality in polynomial time with Algorithm 2, with a complexity in $O(N^3)$ time and in $O(N^2)$ memory space.*

Proof. The formula (16) uses only values $C_{i,j}$ with $j < k$ in Algorithm 2. Induction proves that $C_{i,k}$ has its final value for all $i \in [\![1, N]\!]$ at the end of the for loops from $k = 2$ to K. $C_{N,k}$ is thus at the end of these loops the optimal value of the k-medoids clustering among the N points of E. The backtracking phase searches for the equalities in $C_{i,k} = C_{j'-1,k-1} + c_{j',i}$ to return the optimal clusters $\mathcal{C}_{j',i}$. Let us analyze the complexity. Sorting and indexing the elements of E following Lemma 2 has a complexity in $O(N \log N)$. The computation of the matrix $c_{i,i'}$ has a complexity in $O(N^3)$ thanks to Algorithm 1 and Proposition 2. The construction of the matrix $C_{i,k}$ requires $N \times K$ computations of $\min_{j \in [\![1,i]\!]} C_{j-1,k-1} + c_{j,i}$, which are in $O(N)$, the complexity of this phase is in $O(K.N^2)$. The backtracking phase requires K computations having a complexity in $O(N)$, the complexity is in $O(K.N)$. The bottleneck is the computation of the matrix $c_{i,i'}$ as $K < N$, Hence, the complexity of Algorithm 1 is in $O(N^3)$ time and in $O(N^2)$ memory space because of the initialization phase. $\square$

We note that a similar DP algorithm solves HSS and 1-d k-means, with a similar complexity in $O(K.N^2)$ time and $O(K.N)$ memory space in [2, 22]. In both cases, the time complexity of the DP was improved: time complexity in $O(KN)$ using memory space in $O(N)$ for 1-d k-means in [8], and time complexity for HSS in $O(K.N + N \log N)$ since [3] and in $O(K.(N-K) + N \log N)$ since [12]. Similarly, perspectives are to speed-up the construction of the matrix $C_{i,k}$. However, the complexity in Algorithm 2 is mainly due to the initialization, it reduces the impact of such speed-up.

The practical efficiency can also be improved with a parallel implementation. The parallelization of Algorithm 1, as previously described, is crucial: the initialization phase is indeed the bottleneck for the complexity. The backtracking phase is sequential, but it has the lowest complexity. The second phase, constructing matrix C can also be parallelized. Indeed, once $C_{i,k}$ are computed for all $i \leqslant N$ and a given k, it allows to compute independently all the $C_{i,k+1}$ for $i \leqslant N$. A construction of the matrix C, line by line with k increasing, allows to have K iterations with N independent computations that can be distributed in several processors. This parallel scheme is straightforward to implement in a shared memory environment with OpenMP. For a distributed implementation with MPI, $C_{i,k}$ computations for all $i \leqslant N$ require only $C_{i,k-1}$ in memory for all $i \leqslant N$. At most one N-dimensional array must be stored for each processor, MPI_AllGather operations at each iteration k allows to distribute the results that are required for the next iteration.

6 Bi-objective clustering, how to choose K?

A crucial point for the real life application of clustering is to select an appropriate value of K. A too small value of K can miss that an instance is well-captured with $K+1$ representative clusters, whereas a high value of K gives redundant representatives. The real-life application seeks for the best compromise between the minimization of K, and the minimization of the dissimilarity among the clusters. If for a fixed value of K, K-medoids clustering is solvable in polynomial time, the question is here to analyze the complexity of the bi-objective extension. Actually, the bi-objective extension has the same complexity as the K-medoids clustering problem for a fixed value of K:

Theorem 2. *Let $E = \{x_1, \dots, x_N\}$ a subset of N points of $\mathbb{R}^2$, such that for all $i \neq j$, $x_i \ \mathcal{I} \ x_j$. The bi-objective clustering extension, minimizing conjointly the* *number of clusters k and the k-medoids measure is also solvable to optimality in polynomial time with a complexity in $O(N^3)$ time and in $O(N^2)$ memory space.*

Proof. Using Algorithm 2 with $K = N$, The computation has a complexity of $O(N^3)$, complexity of both the initialization of matrix c and the construction of matrix C. The Pareto front of the bi-objective optimization is included in the unions of $\{(k, C_{k,N})\}$, similarly with the computation of Pareto fronts with the ε-constraint method. Removing the dominated points has a complexity in $O(N)$, negligible compared to $O(N^3)$. This computes the Pareto front, solution of the bi-objective minimizing conjointly the number of clusters and the dissimilarity of clusters with k-medoids. Furthermore, the optimal clusters related to points $\{(k, C_{k,N})\}_k$ can be computed with backtracking operations from $C_{k,N}$ like in the Algorithm 2. Each point computation requiring $O(N^2)$ operations, all the solutions can be computed in $O(N^3)$. □

Theorem 2 allows to compute the whole Pareto front $\{(k, C_{k,N})\}_k$ with the same complexity than only one point of this Pareto front. Searching for good values of k, the elbow technique, graph test, gap test as described in [17], apply to select a good value of k from the Pareto front of couples $\{(k, C_{k,N})\}_k$.

7 Conclusion and perspectives

This paper examined properties of the k-medoids problem in the special case of a discrete set of non-dominated points in a two dimensional Euclidian space. A characterization of optimal clusters is proven with interval clustering. It allows to solve k-medoids to optimality with a dynamic programming algorithm for this special case. A polynomial complexity is proven in $O(N^3)$ time and $O(N^2)$ memory space. The bi-objective extension, minimizing conjointly the number of clusters and the dissimilarity of clusters, is also solvable to optimality in $O(N^3)$ and $O(N^2)$ memory space, which is useful to define how many clusters to use for the real-life application. Parallelization issues are discussed, to speed-up the algorithms in practice.

The complexity in $O(N^3)$ may be a bottleneck to deal with very large Pareto fronts, which open new perspectives. A first perspective is to accelerate the convergence of the exact algorithm, similarly with [12, 8]. A second perspective is to derive specific heuristics to seek for good quality solutions in short resolution time. For both previous issues, parallelization is a complementary perspective to speed-up the resolution of k-medoids clustering for large Pareto fronts.

References

1. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of Euclidean sum-of-squares clustering. Machine learning **75**(2), 245–248 (2009)
2. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In: Proceedings of GECCO 2009. pp. 563–570. ACM (2009)

3. Bringmann, K., Friedrich, T., Klitzke, P.: Two-dimensional subset selection for hypervolume and epsilon-indicator. In: Annual Conference on Genetic and Evolutionary Computation. pp. 589–596. ACM (2014)
4. Dupin, N.: Modélisation et résolution de grands problèmes stochastiques combinatoires: application à la gestion de production d'électricité. Ph.D. thesis, Univ. Lille 1 (2015)
5. Dupin, N., Nielsen, F., Talbi, E.: Clustering in a 2d pareto front: p-median and p-center are solvable in polynomial time. arXiv:1806.02098 pp. 1–24 (2018)
6. Dupin, N., Nielsen, F., Talbi, E.: Dynamic programming heuristic for k-means clustering among a 2-dimensional pareto frontier. 7th International Conference on Metaheuristics and Nature Inspired Computing pp. 1–8 (2018)
7. Ehrgott, M., Gandibleux, X.: Multiobjective combinatorial optimization - theory, methodology, and applications. In: Multiple criteria optimization: State of the art annotated bibliographic surveys, pp. 369–444. Springer (2003)
8. Grønlund, A., Larsen, K.G., Mathiasen, A., Nielsen, J.S., Schneider, S., Song, M.: Fast exact k-means, k-medians and bregman divergence clustering in 1d. arXiv preprint arXiv:1701.07204 (2017)
9. Hsu, W., Nemhauser, G.: Easy and hard bottleneck location problems. Discrete Applied Mathematics **1**(3), 209–215 (1979)
10. Jain, A.: Data clustering: 50 years beyond k-means. Pattern recognition letters **31**(8), 651–666 (2010)
11. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids (1987)
12. Kuhn, T., Fonseca, C.M., Paquete, L., Ruzika, S., Duarte, M.M., Figueira, J.R.: Hypervolume subset selection in two dimensions: Formulations and algorithms. Evolutionary Computation **24**(3), 411–425 (2016)
13. Lloyd, S.: Least squares quantization in PCM. IEEE transactions on information theory **28**(2), 129–137 (1982)
14. Nielsen, F.: Output-sensitive peeling of convex and maximal layers. Information processing letters **59**(5), 255–259 (1996)
15. Nielsen, F.: Introduction to HPC with MPI for Data Science. Springer (2016)
16. Peugeot, T., Dupin, N., Sembely, M.J., Dubecq, C.: MBSE, PLM, MIP and Robust Optimization for System of Systems Management, Application to SCCOA French Air Defense Program. In: Complex Systems Design & Management, pp. 29–40. Springer (2017)
17. Rasson, J.P., Kubushishi, T.: The gap test: an optimal method for determining the number of natural classes in cluster analysis. In: New approaches in classification and data analysis, pp. 186–193. Springer (1994)
18. Saule, E., Baş, E., Çatalyürek, Ü.: Load-balancing spatially located computations using rectangular partitions. Journal of Parallel and Distributed Computing **72**(10), 1201–1214 (2012)
19. Schubert, E., Rousseeuw, P.: Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms. arXiv preprint arXiv:1810.05691 (2018)
20. Sheng, W., Liu, X.: A genetic k-medoids clustering algorithm. Journal of Heuristics **12**(6), 447–466 (2006)
21. Talbi, E.: Metaheuristics: from design to implementation. J. Wiley&Sons (2009)
22. Wang, H., Song, M.: Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. The R journal **3**(2), 29 (2011)
23. Zio, E., Bazzo, R.: 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)