TER - Clustering Algorithms in a 2-dimensional Pareto Front

HE Yuru and XU Yuetian

Universit Paris-Sud, Informatique Master1
yuru.he@u-psud.fr,yuetian.xu@u-psud.fr

Abstract. This paper is motivated by a real life application of multi-objective optimization without preference. Having many incomparable solutions with Pareto optimality, the motivation is to select a small number of representative solutions for decision makers. This paper proves that these clustering problems can be solved to optimality with a unified dynamic programming algorithm. The clustering measures is investigated in this paper for the 2-dimensional case using the specific property that the points to cluster are Pareto optimal in $\mathbb{R}^2$. This paper uses different methods to cluster data, which is Dynamic Programming, Lloyd, Linear Programming, and then compare the advantages and disadvantages of different methods.

Keywords: Clustering algorithms; Pareto frontier; dynamic programming; Lloy; ILP

1 Problem Description

Clustering is a statistical analysis method used to organize raw data into homogeneous silos. Within each cluster, the data is grouped according to a common characteristic. The scheduling tool is an algorithm that measures the proximity between each element based on defined criteria.

In this paper, the representativity measure comes from clustering algorithms, partitioning the N elements into K subsets with a maximal similarity, and giving a representative (i.e. central) element of the optimal clusters.

There are five clustering measures mentioned in this paper: k-means, k-medoids, k-median, discrete k-center,continuous k-center. k-means is a simple and effective measure.In each cluster, there is an averaged center called centroids .It clustering minimizes the sum of squared distance from each item to its nearest averaged center. For k-medoids,in each cluster, there is a medoid, which is a real data item from the data set.k-medoids clustering minimizes the sum of squared distance from each item to its nearest medoids. For k-median ,in each cluster, there is a median.k-median clustering minimizes the sum of distance from each item to its nearest median. For k-center, in each cluster, there is a cluster center.k-center clustering minimizes the maximum distance from each item to its nearest cluster centers.[8]

For the Lloyd's algorithm, it uses iterative refinement heuristics.

Dynamic programming algorithms are often used to solve problems with some optimal properties. There may be many feasible solutions to this type of problem. Each solution corresponds to a value, and we want to find a solution with the best value. The dynamic programming algorithm is similar to the divide-and-conquer method. The basic idea is to decompose the problem to be solved into several sub-problems, first solve the sub-problems, and then get the solution to the original problem from the solutions of these sub-problems.Striving for orderly points, dynamic planning is a new way to think about.

2 Notations

We suppose in this paper having 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:

$$y \ \mathcal{I} \ z \Longleftrightarrow y \prec z \ \text{ or } \ z \prec y \tag{1}$$

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

This property is verified in the applicative context, E being the solution of a bi-objective optimization problem without preference. This applies for exact approaches or population meta-heuristics like evolutionary algorithms and others[1] [**?**].

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

$$d(y,z) = \|y - z\| = \sqrt{\left(y^1 - z^1\right)^2 + \left(y^2 - z^2\right)^2} \tag{3}$$

We define $\Pi_K(E)$, as the set of all the possible partitions of E in K subsets:

$$\Pi_K(E) = \left\{ P \subset \mathcal{P}(E) \,\middle|\, \forall p, p' \in P,\ \ p \cap p' = \emptyset \text{ and } \bigcup_{p \in P} = E \text{ and } \operatorname{card}(P) = K \right\} \tag{4}$$

Defining a cost function f for each subset of E to measure the dissimilarity, the clustering problem is written as following optimization problem:

$$\min_{\pi \in \Pi_K(E)} \sum_{p \in \pi} f(P) \tag{5}$$

K-means clustering is a combinatorial optimization problems indexed by $\Pi_K(E)$. K-means clustering minimizes the sum for all the K clusters of the average distances from the points of the clusters to the centroid. Mathematically, this can be written as:

$$f_{means}(P) = \frac{1}{\operatorname{card}(p)} \sum_{x \in p} \left\| x - \frac{1}{\operatorname{card}(p)} \sum_{y \in p} y \right\|^2 \tag{6}$$

K-medoids clustering minimizes the sum of squared distance from each item to its nearest medoids.Mathematically, this can be written as:

$$f_{medoids}(P) = \frac{1}{\operatorname{card}(p)} \min_{y \in p} \sum_{x \in p} \|x - y\|^2 \tag{7}$$

K-median clustering minimizes the sum of distance from each item to its nearest median.Mathematically, this can be written as:

$$f_{median}(P) = \min_{y \in p} \sum_{x \in p} \|x - y\| \tag{8}$$

Discrete K-center can be written as:

$$f_{ctr}^{D}(P) = \min_{y \in p} \max_{x \in p} \|x - y\| \tag{9}$$

Continous K-center can be writtern as:

$$f_{ctr}^{C}(P) = \min_{y \in \mathbb{R}^2} \max_{x \in p} \|x - y\| \tag{10}$$

Industrial applications of Pareto Front in multi-objective optimization: system engineering, design of industrial systems. In the case of dimension 2, minimizing 2 objectives, we can define Pareto fronts more easily:

A discrete 2-dimensional Pareto Front is defined here as a set E of N points in $\mathbb{R}^2$, indexed with $E = (x_k, y_k)_{k \in [\![1,N]\!]}$ such that $k \in [\![1,N]\!] \mapsto x_k$ is increasing and $k \in [\![1,N]\!] \mapsto y_k$ is decreasing.

3 Dynamic Programming Algorithm

In computing, dynamic programming is an algorithmic method for solving optimization problems. Dynamic programming is about solving a problem by breaking it down into subproblems, then solving the subproblems, from the smallest ones to the big ones by storing intermediate results.

Conjecture and the polynomial computations of all the $c_{i,i'}$ with $i < i'$ allows to derive a dynamic programming algorithm. Defining $C_{i,k}$ as the optimal cost of the k-means clustering with k cluster among points $[\![1,i]\!]$ for all $i \in [\![1,N]\!]$ and $k \in [\![1,K]\!]$, we have following induction relation:

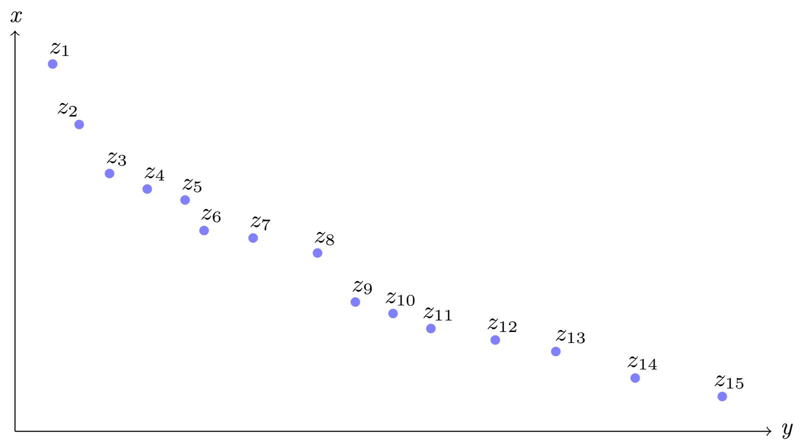

Fig. 1. Illustration of the indexation implied by Proposition in a 2-d Pareto front

$$\forall i \in [\![1, N]\!],\ \forall k \in [\![2, K]\!], \quad C_{i,k} = \min_{j \in [\![1,i]\!]} C_{j-1,k-1} + c_{j,i} \tag{11}$$

This last relation use the convention that $C_{0,k} = 0$ for all $k \geqslant 0$. These relations allow to compute the optimal values of $C_{i,k}$ for all $i \in [\![1, N]\!]$ and $k \in [\![1, K]\!]$. The optimal partition is computed with backtracking.

Algorithm 1: DP interval clustering in a 2d-Pareto Front

```
compute cost matrix $c_{i,j}$ for all $(i,j) \in [\![1,N]\!]^2$

   for $i = 1$ to $N$ //Construction of the matrix $C$
     set $C_{i,k} = c_{1,i}$ // case $k = 1$ treated separately
     for $k = 2$ to $K$
       if $(C_{i,k} \geq C_{i-1,k-1})$
         set $C_{i,k} = \min_{j \in [\![1,i]\!]} C_{j-1,k-1} + c_{j,i}$
     end for
   end for
return $C_{N,K}$ the optimal cost

   initialize $i = N$ and $P = nil$ //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 the partition $\mathcal{P}$ giving the cost $C_{N,K}$
```

4 Lloyd Algorithm

Simply speaking, K-Means clustering is an algorithm to classify or to group your objects based on attributes or features, into K number of groups. K is a positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster center. Thus, the purpose of K-means clustering is to classify the data.

K-Means works better when the resulting clusters are dense and the differences between clusters and clusters are significant. For large data sets, K-Means is relatively scalable and efficient, its

Clustering Problem	Complexity	Complexity	Complexity
K-means	O(i'-i)	O(N^3)	O(N^3)
K-medoids	$O((i'-i)^2)$	O(N^4)	O(N^4)
K-median	$O((i'-i)^2)$	O(N^4)	O(N^4)
discr.K-center	O(i'-i)	O(N^3)	O(N^3)
cont.K-center	O(1)	O(N^2)	O($K.N^2$)

Fig. 2. Complexities with naive computation of cluster costs

complexity is $O(nkt)$, n is the number of objects, k is the number of clusters, and t is the number of iterations, usually $k \ll n$, and $t \ll n$, so the algorithm often ends with a local optimum.

The biggest problem with K-Means is that it is required to give the number of k first. The choice of k is generally based on empirical values and multiple experimental results. For different data sets, the value of k has no reference. In addition, K-Means is sensitive to outlier data, and a small amount of noise data can have a significant impact on the average.

The most common implementation of the K-Means algorithm is the Lloyd's algorithm using iterative refinement heuristics.[9]

Given the number of divisions k. Create an initial partition, randomly selecting k objects from the dataset, each object initially representing a cluster center. For other objects, calculate their distance from each cluster center and divide them into the nearest cluster.

Using iterative relocation techniques, attempts to improve partitioning by moving objects between partitions. The so-called relocation technique is to recalculate the average of the clusters when new objects are added to the cluster or existing objects leave the cluster, and then the objects are reallocated. This process is repeated until the objects in each cluster no longer change.

Algorithm 2: Lloyd for k-means

```
Randomly generate n center points.
   for $t = 1, 2....T$ //T is the number of cycles
      for every points $x_i$
         for $i = 1$ to $N$
            calculate $dist(x_i, center)$//Divide $x_i$ to the nearest center
      end for
update all the center
 end for
```

5 Integer Linear Programming

Within the Integer Linear Programming (ILP) framework, we can formulate the k-median problem for example. A first ILP formulation defines binary variables $x_{i,j} \in \{0,1\}$ and $y_j \in \{0,1\}$. $x_{i,j} = 1$ if and only if the customer i is assigned to the depot j. $y_j = 1$ if and only if the point f_j is chosen as a depot. Following ILP formulation expresses the k-median problem:[2,3]

$$\begin{aligned} min_{x,y} \quad & \sum_{j=1}^{n}\sum_{i=1}^{n} d_{i,j} x_{i,j} \\ s.t: \quad & \sum_{j=1}^{n} y_j = p \\ & \sum_{j=1}^{n} x_{i,j} = 1, \forall i \in [\![1, N]\!] \\ & x_{i,j} \le y_j, \forall (i,j) \in [\![1, N]\!]^2, \\ \forall i,j, \quad & x_{i,j}, y_j \in 0,1 \end{aligned} \tag{12}$$

The k-median problem was originally a logistic problem, having a set of customers and defining the places of depots in order to minimize the total distance for customers to reach the closest depot. We give here the general form of the k-median problem. Let N be the number of clients, called $c_1, c_2, ..., c_N$, let M be the number of potential sites or facilities,called $f_1, f_2, ..., f_M$,and let $d_{i,j}$ be the distance from c_i to f_j. The k-median problem consists of opening p facilities and assigning each client to its closest open facility, in order to minimize the total distance. We note that in the general k-median problem, the graph of the possible assignments is not complete, which can be modeled with $d_{i,j} = +\propto$. In our application,the graph is complete,the points $f_1, f_2, ..., f_M$ are exactly $c_1, c_2, ..., c_N$ and $d_{i,j}$ is the Euclidian distance in $\mathbb{R}^{\not{L}}$.

The algorithm is as follow: Our application is a discrete k-center problem, the points $f_1, f_2, ..., f_M$

Algorithm 3: Computation of matrix $c_{i,i'}$ for the k-median prem

```
//Initialization phase.
define matrix c with $c_{i,i'} = 0$ for all $c_{i,i'} \in [\![1; N]\!]^2 with$ i $\leq i'$
define matrix d with $d_{i,i'} = 0$ for all $c_{i,l,i'} \in [\![1; N]\!]^3 with$ i $\leq l \leq i'$
  for $l = 1$ to $N$ //Consider subset of cardinal $l$
    for $i = 1$ to $N - l$
      for $k = i$ to $i + l$
        $d_{i,k,i+l} = d_{i,k,i+l-1} + \parallel x_{i+1} - x_k \parallel$
      end for
      Compute $c_{i,i+l} = \min_{k \in [i,i+l]} d_{i,k,i+l}$
    end for
  end for
return $C_{i,i'}$
```

being exactly $c_1, c_2, ..., c_N$. Similarly with the k-median problem, following ILP formulation models the discrete k-center problem:

$$\begin{aligned} & min_{x,y,z} \quad z_i \\ s.t: \quad & \sum_{j=1}^{n} \sum_{i=1}^{n} \leq z, \forall i \in [\![1, n]\!] \\ & \sum_{j=1}^{n} y_j = p \\ & \sum_{j=1}^{n} x_{i,j} = 1 \\ & x_{i,j} \leq y_j, \forall (i,j) \in [\![1, n]\!]^2 \\ \forall i, j, \quad & x_{i,j}, y_j \in 0, 1 \end{aligned} \tag{13}$$

Following ILP formulation models the continuous k-center problem:

$$\begin{aligned}
& min_{x,y,z} \quad z_i \\
s.t: \quad & \sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{i=1}^{n} d_{i,j} * (x_{k,i} + x_{k,j} - 1) \leq z_i, \forall k \in [\![1, n]\!] \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j} = 1 \forall i \in [\![1, n]\!] \\
& \sum_{j=1}^{n} y_j = k \\
& x_{i,j} \leq y_j, \forall (i,j) \in [\![1, n]\!]^2 \\
\forall i, j, \quad & x_{i,j}, y_j \in 0, 1
\end{aligned} \tag{14}$$

We can re-optimize the algorithm with Pareto Front.

In the experiment, in order to use the idea of ILP, we used CPLEX. CPLEX is a linear program solver. As such, it rests soon a successful implementation of the primal simplex. It also has simplex dual and network simplex. It can also solve mixed linear programs, in combining simplex, branch and bound and section generation. Recently, IT integrates also a technique based on interior points and little to deal with quadratic problems. Currently, CPLEX is one of the most powerful solvers available, if not the most performance. It can thus deal with problems containing tens of thousands of variables and several hundreds of thousands of constraints. For mixed problems, the limit is significantly lower, but it depends greatly on the type of problems and the model applied.[4]

6 Experimental Result Data

According to several possibilities to accelerate the DP algorithm in practice. The results of comparison of computation time and number of operations required for several variants are gained.

We use cplex, Lloyd algorithm and Dynamic Programming algorithm to test these different clustering methods. The test was performed using different data sets of different numbers of N.

6.1 Time Comparison

When we test k-median, we tested 9 different cases where n equals 10, 50 and 100. It can be found that the DP algorithm takes much longer than the Lloyd algorithm, and the Lloyd algorithm is greater than using cplex[Fig.3,4].

	Cplex	Lloyd	DP
dataAlea2_50_ex1	0.050s	0.002s	1.591s
dataAlea2_50_ex2	0.050s	0.001s	1.872s
dataAlea2_50_ex3	0.050s	0.003s	2.427s
dataAlea2_100_ex1	0.250s	0.004s	8.334s
dataAlea2_100_ex2	0.240s	0.002s	8.028s
dataAlea2_100_ex3	0.260s	0.004s	7.886s

Fig. 3. Time comparison 1 with k-median

Next we continue to test two k-center algorithms. In the discrete k-center algorithm, we tested with Cplex and DP and found that the results are very similar. Next the Continuous k-center algorithm, We tested three methods using Cplex, Cplex-Pareto and DP, and found that Cplex-Pareto has the fastest operation, followed by DP and the last one is the Cplex[Fig.5,6].

When we test k-medoids, we get the results that the DP algorithm takes much longer than the Lloyd algorithm, and the Lloyd algorithm is greater than using cplex[Fig.7].

Regardless of the method, when the amount of data becomes larger, the increase in time is not linear, and the program needs multiple times to process the data. Relatively speaking, the Lloyd algorithm's time increase is the most acceptable, so the Lloyd algorithm is more advantageous when dealing with large amounts of data.

	Cplex	Lloyd	DP
dataConvex_3_10_2_10_ex1	0.010s	0.001s	2.634s
dataConvex_3_10_2_10_ex2	0.010s	0.001s	1.709s
dataConvex_3_10_2_10_ex3	0.010s	0.001s	1.011s
dataConvex_3_10_2_10_ex4	0.010s	0.001s	1.480s
dataConvex_3_10_2_50_ex1	0.050s	0.004s	1.542s
dataConvex_3_10_2_50_ex2	0.060s	0.003s	1.325s
dataConvex_3_10_2_50_ex3	0.050s	0.003s	1.151s
dataConvex_3_10_2_50_ex4	0.050s	0.004s	1.973s
dataConvex_3_10_2_100_ex1	0.280s	0.014s	8.027s
dataConvex_3_10_2_100_ex2	0.290s	0.005s	8.396s
dataConvex_3_10_2_100_ex3	0.310s	0.014s	7.668s
dataConvex_3_10_2_100_ex4	0.310s	0.008s	8.033s

Fig. 4. Time comparison 2 with k-median

	Cplex	DP
dataAlea2_50_ex1	1.030s	1.331s
dataAlea2_50_ex2	1.750s	1.769s
dataAlea2_50_ex3	2.490s	1.767s
dataAlea2_100_ex1	238.950s	2.030s
dataAlea2_100_ex2	19.840s	1.576s
dataAlea2_100_ex3	170.620s	1.419s

Fig. 5. Time comparison with discrete k-center

	Cplex	Cplex-Pareto	DP
dataAlea2_50_ex1	3.490s	0.090s	1.107s
dataAlea2_50_ex2	3.790s	0.060s	1.484s
dataAlea2_50_ex3	4.300s	0.090s	1.100s
dataAlea2_100_ex1	449.720s	0.680s	1.843s
dataAlea2_100_ex2	143.780s	0.640s	1.308s
dataAlea2_100_ex3	724.630s	0.840s	2.013s

Fig. 6. Time comparison with continuous k-center

	Cplex	Lloyd	DP
dataAlea2_50_ex1	0.060s	0.007s	1.705s
dataAlea2_50_ex2	0.050s	0.002s	1.660s
dataAlea2_50_ex3	0.050s	0.004s	1.705s
dataAlea2_100_ex1	0.250s	0.010s	14.751s
dataAlea2_100_ex2	0.250s	0.006s	14.890s
dataAlea2_100_ex3	0.230s	0.004s	15.353s

Fig. 7. Time comparison with k-medoids

6.2 Quality of solution

For a more intuitive judgment of the quality of the method. We process the calculated cost. First find out the minimum cost by different methods. Each cost minus the minimum cost, then divided by the minimum cost. The value thus obtained, we use this value to judge the quality of the method. The closer this value is to 0, the better the method.

$$v = ((cost - cost\_min)/cost\_min) * 100\%$$

We can intuitively see that the method with a v equal to 0 is the best method for this set of data.

	Cplex	Lloyd	DP
dataAlea2_50_ex1	20.736	4.060	0
dataAlea2_50_ex2	3.444	8.341	0
dataAlea2_50_ex3	31.701	2.223	0
dataAlea2_100_ex1	1.855	12.734	0
dataAlea2_100_ex2	26.308	2.708	0
dataAlea2_100_ex3	7.023	1.958	0
dataConvex_3_10_2_10_ex1	0	0	0
dataConvex_3_10_2_10_ex2	2.013	2.013	0
dataConvex_3_10_2_10_ex3	4.892	0	0
dataConvex_3_10_2_10_ex4	32.673	26.463	0
dataConvex_3_10_2_50_ex1	14.270	3.981	0
dataConvex_3_10_2_50_ex2	15.497	13.854	0
dataConvex_3_10_2_50_ex3	17.261	0.353	0
dataConvex_3_10_2_50_ex4	17.431	0.742	0
dataConvex_3_10_2_100_ex1	7.435	0.820	0
dataConvex_3_10_2_100_ex2	7.474	3.979	0
dataConvex_3_10_2_100_ex3	7.578	2.527	0
dataConvex_3_10_2_100_ex4	7.629	4.041	0

Fig. 8. Cost quality with k-median

	Cplex	Lloyd	DP
dataAlea2_50_ex1	0	14.051	14.051
dataAlea2_50_ex2	5.486	5.422	0
dataAlea2_50_ex3	0	6.287	6.287
dataAlea2_100_ex1	5.109	3.679	0
dataAlea2_100_ex2	231.643	0	0
dataAlea2_100_ex3	38.877	0	0
dataConvex_3_10_2_10_ex1	0	69.372	0
dataConvex_3_10_2_10_ex2	5.626	0	2.447
dataConvex_3_10_2_10_ex3	6.706	0	4.160
dataConvex_3_10_2_10_ex4	35.813	0	0
dataConvex_3_10_2_50_ex1	81.102	42.500	0
dataConvex_3_10_2_50_ex2	50.0368	4.026	0
dataConvex_3_10_2_50_ex3	59.604	15.767	0
dataConvex_3_10_2_50_ex4	103.375	73.849	0
dataConvex_3_10_2_100_ex1	61.329	0.767	0
dataConvex_3_10_2_100_ex2	52.511	0	0
dataConvex_3_10_2_100_ex3	52.394	21.598	0
dataConvex_3_10_2_100_ex4	55.476	8.631	0

Fig. 9. Cost quality with k-medoids

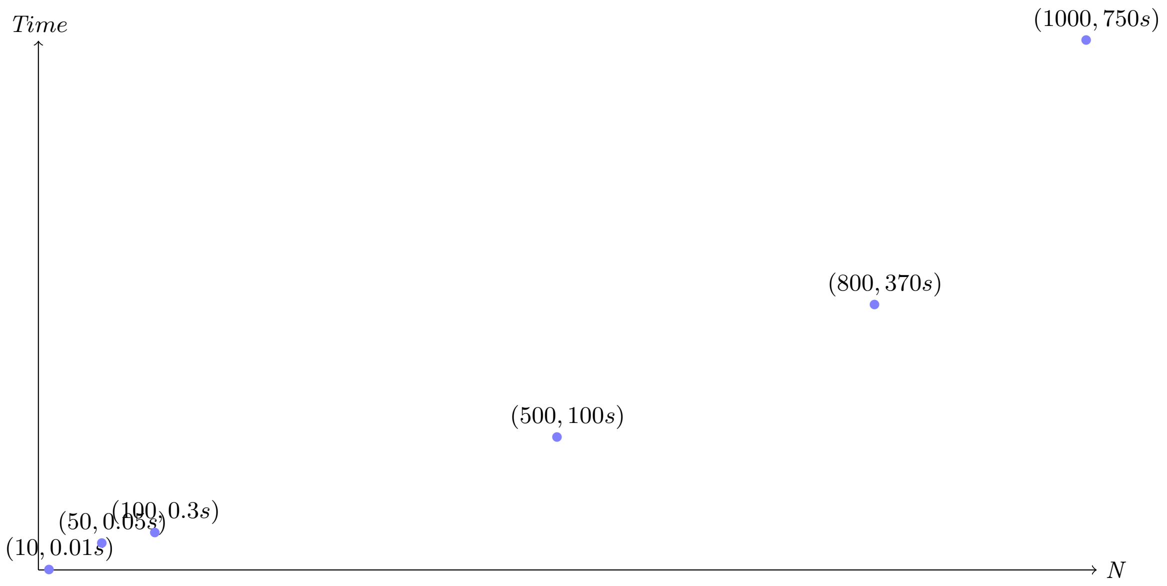

Fig. 10. The change of time with the number of points in Cplex

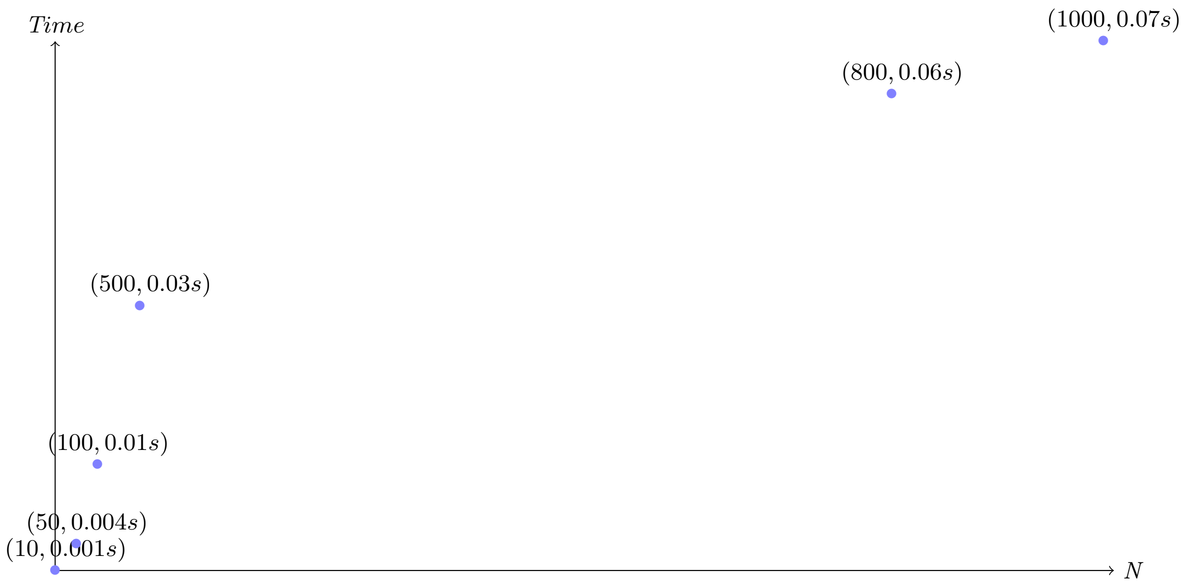

Fig. 11. The change of time with the number of points in Lloyd

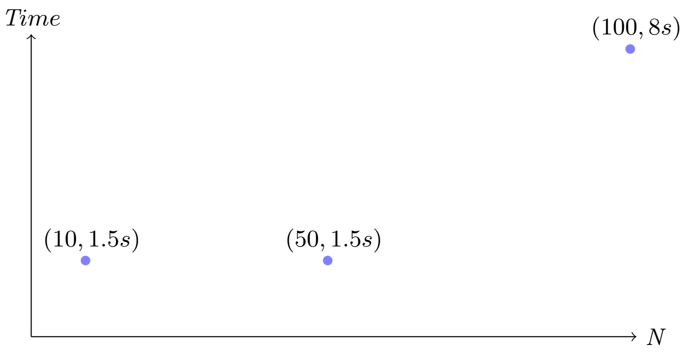

Fig. 12. The change of time with the number of points in DP

6.3 Conclusion

After completing all the experimental data analysis, based on the results, we can draw the following summary: The advantage of the Dynamic programming algorithm is that for the ordered data, the optimal solution can generally be obtained. But the disadvantage is that once the result is not good for the completely random number sequence, and if the amount of data is too large, it cannot be realized.

The advantage of the Lloyd algorithm is that the data requirements are low, and the results are relatively good. At the same time, the data with a large amount of data can be processed, and the calculation speed is fast. But the downside is that because in the first step, the selection of the center point is random, then for some very discrete data good performance is not obtained. For example, in a group of data, a few points are scattered, and the remaining points are relatively concentrated. The advantage of Cplex is that when dealing with large amounts of data, you can manually set the time limit and the code is simpler. But the disadvantage is that when the amount of data is large, the software can't handle it. And the cluster quality is poor.

References

1. E-G. Talbi, M. Basseur, A. Nebro, and E. Alba. Multi-objective optimization using metaheuristics: non-standard algorithms. International Transactions in Operational Research, 19(1-2):283305, 2012.
2. C. Beltran, C. Tadonki, and J. P. Vial. Solving the p-median problem with a semilagrangian relaxation. Computational Optimization and Applications, 35(2):239260, 2006.
3. A. Santos. Solving large p-median problems using a lagrangean heuristic. 2009.
4. P. Avella, A. Sassano, and I. Vasilev. Computational study of large-scale p-median problems. Mathematical Programming, 109(1):89114, 2007.
5. C. Beltran, C. Tadonki, and J. P. Vial. Solving the p-median problem with a semilagrangian relaxation. Computational Optimization and Applications, 35(2):239260, 2006.
6. S. Dasgupta. The hardness of k-means clustering. Department of Computer Science and Engineering, University of California, San Diego, 2008.
7. S. Elloumi. A tighter formulation of the p-median problem. Journal of combinatorial optimization, 19(1):6983, 2010.
8. A. Grnlund, K. Larsen, A. Mathiasen, J. Nielsen, S. Schneider, and M. Song. Fast exact k-means, k-medians and Bregman divergence clustering in 1d. arXiv preprint arXiv:1701.07204, 2017.
9. L. Guo-Hui and G. Xue. K-center and k-median problems in graded distances. Theoretical computer science, 207(1):181192, 1998.
10. O. Kariv and S. Hakimi. An algorithmic approach to network location problems. ii: The p-medians. SIAM Journal on Applied Mathematics, 37(3):539560, 1979.
11. N. Megiddo and A. Tamir. New results on the complexity of p-centre problems. SIAM Journal on Computing, 12(4):751758, 1983.
12. N. Megiddo and K. Supowit. On the complexity of some common geometric location problems. SIAM journal on computing, 13(1):182196, 1984.

A Appendix: Data result

	Cplex	Lloyd	DP
dataAlea2_50_ex1	2058.093	1773.838	1704.62
dataAlea2_50_ex2	2021.817	2117.542	1954.52
dataAlea2_50_ex3	3096.82	2403.684	2351.41
dataAlea2_100_ex1	5824.758	6446.903	5718.7
dataAlea2_100_ex2	3003.123	2442.001	2377.62
dataAlea2_100_ex3	6972.896	6642.886	6515.32

Fig. 13. Cost comparison 1 with k-median

	Cplex	Lloyd	DP
dataConvex_3_10_2_10_ex1	1.225714	1.2257136	1.22571
dataConvex_3_10_2_10_ex2	2.026535	2.0265349	1.98655
dataConvex_3_10_2_10_ex3	3.219636	3.0694910	3.07206
dataConvex_3_10_2_10_ex4	3.06849	2.92485976	2.31282
dataConvex_3_10_2_50_ex1	24.53523	22.325871	21.4712
dataConvex_3_10_2_50_ex2	22.48331	22.1634272	19.4666
dataConvex_3_10_2_50_ex3	20.54524	17.5826658	17.5209
dataConvex_3_10_2_50_ex4	20.62211	17.6912253	17.5610
dataConvex_3_10_2_100_ex1	43.99194	41.283480	40.9474
dataConvex_3_10_2_100_ex2	44.44086	42.9955266	41.3502
dataConvex_3_10_2_100_ex3	43.21486	41.1860666	40.1709
dataConvex_3_10_2_100_ex4	43.58614	42.1334181	40.4968

Fig. 14. Cost comparison 2 with k-median

	Cplex	DP
dataAlea2_50_ex1	436.3841	421.085
dataAlea2_50_ex2	440.925	378.452
dataAlea2_50_ex3	463.8659	416.044
dataAlea2_100_ex1	493.423	451.817
dataAlea2_100_ex2	477.7782	323.907
dataAlea2_100_ex3	594.3583	456.476

Fig. 15. Cost comparison with discrete k-center

	Cplex	Cplex-Pareto	DP
dataAlea2_50_ex1	408.3283	398.0015	421.085
dataAlea2_50_ex2	378.4519	185.7544	378.452
dataAlea2_50_ex3	431.0037	416.729	416.044
dataAlea2_100_ex1	296,6047	164.3702	451.817
dataAlea2_100_ex2	400.5863	323.472	323.907
dataAlea2_100_ex3	499,9950	498.4939	456.476

Fig. 16. Cost comparison with continuous k-center

	Cplex	Lloyd	DP
dataAlea2_50_ex1	233207.2	265975.0147	265975
dataAlea2_50_ex2	270378.0	270215.5484	256317
dataAlea2_50_ex3	187504.5	199292.1990	199292
dataAlea2_100_ex1	894084.8	850628.3129	850628
dataAlea2_100_ex2	553763.9	166976.2016	166976
dataAlea2_100_ex3	1051562	757190.92736	757191

Fig. 17. Cost comparison with k-medoids

	Cplex	Cplex-Pareto	DP
dataAlea2_50_ex1	408.3283	398.0015	421.085
dataAlea2_50_ex2	378.4519	185.7544	378.452
dataAlea2_50_ex3	431.0037	416.729	416.044
dataAlea2_100_ex1	296,6047	164.3702	451.817
dataAlea2_100_ex2	400.5863	323.472	323.907
dataAlea2_100_ex3	499,9950	498.4939	456.476
dataConvex_3_10_2_10_ex1	0.5907992	0.5907992	0.590799
dataConvex_3_10_2_10_ex2	0.9944741	0.9944741	0.993272
dataConvex_3_10_2_10_ex3	1.334924	1.334924	1.53602
dataConvex_3_10_2_10_ex4	1.255716	1.255716	1.15641
dataConvex_3_10_2_50_ex1	2.929918	1.569714	2.77596
dataConvex_3_10_2_50_ex2	2.861603	1.540522	2.60684
dataConvex_3_10_2_50_ex3	2.526753	1.699532	2.52675
dataConvex_3_10_2_50_ex42.74341	1.594426	2.74324	

Fig. 18. Cost comparison with continuous k-cente

	Cplex	Cplex-Pareto	DP
dataAlea2_50_ex1	2.594663588	0	5.799853
dataAlea2_50_ex2	103.7377849	0	103.7378
dataAlea2_50_ex3	3.595701416	0.16464605	0
dataAlea2_100_ex1	1804391.933	0	174.8777
dataAlea2_100_ex2	23.83955953	0	0.134478
dataAlea2_100_ex3	1095236.885	9.20484319	0
dataConvex_3_10_2_10_ex1	3.38525	3.3852	0
dataConvex_3_10_2_10_ex2	0.121024251	0.12102425	0
dataConvex_3_10_2_10_ex3	0	0	15.06423
dataConvex_3_10_2_10_ex4	8.587438711	8.58743871	0
dataConvex_3_10_2_50_ex1	86.65298265	0	76.84495
dataConvex_3_10_2_50_ex2	85.75541278	0	69.21797
dataConvex_3_10_2_50_ex3	48.67345834	0	48.67328
dataConvex_3_10_2_50_ex4	72.0625479	0	72.05189

Fig. 19. Cost quality comparison with continuous k-cente