ELSEVIER

Discrete Mathematics 245 (2002) 93–105

DISCRETE MATHEMATICS

www.elsevier.com/locate/disc

Polynomial algorithms for nested univariate clustering ☆

Pierre Hansen[a,*], Brigitte Jaumard[b], Bruno Simeone[c]

[a] *GERAD and École des Hautes Études Commerciales, Ecole Polytechnique Montréal, 5255 Ave. Decelles Montreal, Quebec, Canada H3T 1V6*

[b] *GERAD and École Polytechnique de Montréal, 5255 Ave. Decelles Montreal, Quebec, Canada H3T 1V6*

[c] *Department of Statistics, University "La Sapienza", Rome, Italy*

Received 17 September 1998; revised 11 July 2000; accepted 16 October 2000

Abstract

Clique partitioning in Euclidean space $\mathbb{R}^n$ consists in finding a partition of a given set of N points into M clusters in order to minimize the sum of within-cluster interpoint distances. For $n=1$ clusters need not consist of consecutive points on a line but have a nestedness property. Exploiting this property, an $O(N^5M^2)$ dynamic programming algorithm is proposed. A $\theta(N)$ algorithm is also given for the case $M=2$. © 2002 Published by Elsevier Science B.V.

Keywords: Clustering; Clique; Polynomial algorithm

1. Introduction

Let $O=\{O_1,O_2,\ldots,O_N\}$ denote a set of N entities and $D=(d_{k\ell})$, $k,\ell=1,2,\ldots,N$, a matrix of real numbers such that $d_{kk}=0$, $d_{k\ell}=d_{\ell k}$ and $d_{k\ell}\geqslant 0$ for all k and ℓ. Partitioning problems of cluster analysis [8–11,13] consist in finding a partition $P_M=\{C_1,C_2,\ldots,C_M\}$ of O into M *clusters* $C_1,C_2,\ldots,C_M$, i.e., $C_j\neq\emptyset$, $C_i\cap C_j=\emptyset$ for $i,j\neq i=1,2,\ldots,M$ and $\bigcup_{j=1}^{M}C_j=O$, which minimizes (or maximizes) an objective function $F(C_1,C_2,\ldots,C_M)$. The $d_{k\ell}$ are called *dissimilarities* and express the magnitude of the differences observed between entities O_k and O_ℓ. The objective function

☆ We are grateful to the referees for their careful reading and their many suggestions which led to substantial improvements in the paper's presentation. Research supported by ONR grant N00014-95-1-0917, NSERC (Natural Scientific Research and Engineering Council of Canada) grants GPO105574 and GPO036426, FCAR (Fonds pour la Formation de Chercheurs et l'Aide à la Recherche) grant 95ER1048 and done during visits of the first two authors at University "La Sapienza", Rome, of the first author at École Polytechnique Fédérale de Lausanne and of the second author at Université de Fribourg.

* Corresponding author. Tel.: +1-514-340-6053-5675; fax: +1-514-3405665.

E-mail address: pierreh@crt.umontreal.ca (P. Hansen).

0012-365X/02/$ - see front matter
PII: S0012-365X(01)00135-2

$F(C_1, C_2, \ldots, C_M)$ expresses the desire to obtain a partition with clusters which are *homogeneous* and/or *well separated*. These objectives can be made mathematically precise in many ways, thus giving rise to a large variety of partitioning problems.

In the general case, such problems are usually NP-hard [3,7,17]. (A notable exception is maximizing the *split* of a partition, or smallest dissimilarity between entities in different clusters, which can be solved in $\theta(N^2)$ time by the single-linkage algorithm [6,4].) Particular cases are, of course, often easier.

We consider in this paper univariate clustering with dissimilarities equal to Euclidean distance (or, in other words, Univariate Euclidean Clique Partitioning (UECP)). Then the entities $O_k, O_\ell \ldots$ are points $x_k, x_\ell \ldots$ on the line and $d_{k\ell} = |x_k - x_\ell|$. Moreover, we assume that entities are at M different points or more. This implies that in any optimal partition no cluster is empty. For many objective functions, clusters in an optimal partition, have the *string* or *consecutiveness* property [15]; they consist of consecutive points along this line. When this property holds, it is often the case that the contribution to the objective function value of a cluster bounded by entities at x_i and x_j can be computed in linear time and updated in constant time when a point is added or removed at an end of the cluster. Then a straightforward $O(N^2)$ dynamic programming algorithm applies [1,15]. Low order polynomial algorithms may also be obtained for several more complicated objective functions and for problems with bounds on the clusters size [5,14]. Moreover, for some criteria better results are at hand: e.g. exploiting the recent $O(n \log\log n)$ algorithm of Thorup [16] for sorting n numbers on a RAM machine within Gower and Ross' [6] single linkage algorithm leads to an $O(N \log\log N)$ algorithm for maximizing the split.

However, the string property is not always satisfied for univariate Euclidean clustering. The *clique partitioning* problem consists in minimizing the sum of within cluster distances between pairs of points:

$$F(C_1, C_2, \ldots, C_M) = \sum_{j=1}^{M} d(\ell_j),$$

where

$$d(C_j) = \sum_{h,\ell:\ O_k, O_\ell \in C_j,\ x_k < x_\ell} d_{k\ell}.$$

The following example, from (2), shows the string property does not hold for this objective.

Example 1. $N = 5,\ x_1 = 0,\ x_2 = x_3 = x_4 = 1,\ x_5 = 2,\ M = 2$. The only optimal partition is $P_2^* = \{\{x_1, x_5\}\{x_2, x_3, x_4\}\}$ with $F(C_1^*, C_2^*) = 2$.

A weaker condition does hold. Let $H(C_j)$ denote the convex hull of $C_j,\ j = 1, 2, \ldots, M$. Call the partition $P_M = \{C_1, C_2, \ldots, C_M\}$ *nested* if

$$\forall_{i,j},\ i \neq j: \quad C_j \cap H(C_i) = \emptyset \quad \text{or} \quad C_i \cap H(C_j) = \emptyset,$$

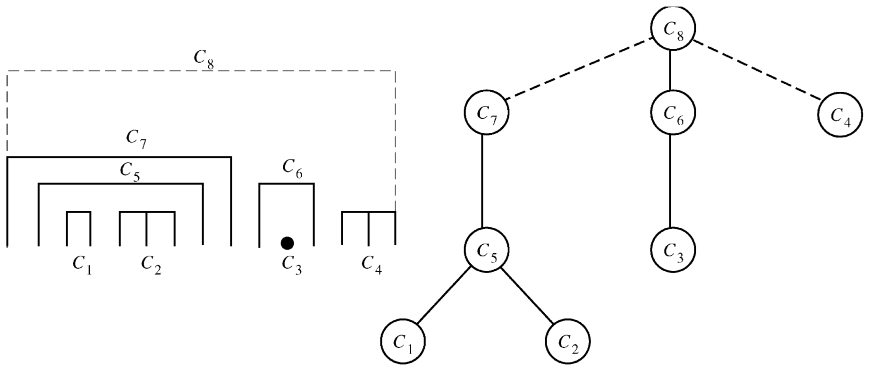

Fig. 1. A nested partition and its encompassment tree.

or, equivalently, if the clusters C_j can be renumbered such that

$$\forall_{i,j},\ i<j:\quad C_j\ \cap H(C_i)=\emptyset.$$

Boros and Hammer [2] prove:

Theorem 2. *Every optimal partition of UECP is nested.*

Another proof is given, and some further weaker properties of univariate partitions are explored in [12].

By definition a cluster C_i with $H(C_i)=[x_k,x_\ell]$ is *encompassed* by a cluster C_j with $H(C_j)=[x_r,x_s]$ if $x_r<x_k$. This implies $x_\ell<x_s$. The encompassment relation will be denoted by $C_i\sqsubset C_j$.

The encompassment relation is irreflexive, asymmetric and transitive. Furthermore, we have the following *Triangular Encompassment* (TE) property:

For any three district clusters C_i, C_j and C_k, if $C_k\sqsubset C_i$ and $C_k\sqsubset C_j$ then either $C_i\sqsubset C_j$ or $C_j\sqsubset C_i$.

The proof is easy and omitted here.

Due to the TE property the Haase diagram of encompassment is a rooted forest, or if a dummy cluster encompassing all other is added, a rooted tree (*called encompassment tree*, see Fig. 1).

Clique partitioning is NP-hard in the general case [3,17]. The complexity of univariate Euclidean clique partitioning was left as an open question in [2]. It is the main purpose of this paper to answer this question, in Section 2, by providing a polynomial algorithm. This algorithm exploits the nestedness property, using dynamic programming, and has a complexity in $O(N^5M^2)$. Furthermore, a $\theta(N)$ algorithm is provided in Section 3 for the case $M=2$.

It is well known that in hierarchical agglomerative clustering an initial partition with each cluster consisting of a single entity is first considered and then pairs of clusters are merged in order to minimize at each step the increase (or decrease) in objective

function value. Such a scheme may be viewed as a greedy algorithm. It leads to a hierarchy of partitions containing jointly $2N-1$ clusters C_j such that

$$\forall_{i,j},\ i \neq j: \quad C_j \cap C_i = \emptyset \ \text{ or } \ C_i \subset C_j \ \text{ or } \ C_j \subset C_i,$$

or, equivalently, the C_j can be renumbered such that

$$\forall_{i,j},\ i < j: \quad C_i \subset C_j \ \text{ or } \ C_j \cap C_i = \emptyset.$$

Hence these clusters are *nested*, but in a different sense than that of [2]. One might wonder if agglomerative hierarchical clustering solves univariate clique partitioning optimally. The following example shows it is not the case.

Example 3. $N=8,\ x_1=0,\ x_2=4,\ x_3=x_4=x_5=x_6=x_7=5,\ x_8=6,\ x_9=9,\ M=3.$ Agglomerative hierarchical clustering gives a partition $P_3=\{\{x_1\}\{x_2,x_8,x_9\}\{x_3,x_4,x_5,x_6,x_7\}\}$ with a value $F(C_1,C_2,C_3)=10$. The only optimal partition is $P_3^*=\{\{x_1,x_2\},\{x_3,x_4,x_5,x_6,x_7\},\{x_8,x_9\}\}$ with $F(C_1^*,C_2^*,C_3^*)=7$.

In the optimal solutions of Examples 1 and 3 above, entities at the same point on the line belong to the same cluster. This is always possible.

Proposition 4. *In at least one optimal solution UECP all entities at the same point belong to the same cluster.*

Proof. Suppose that point $x=x_j$ has q replications and that, in some optimal partition and w.l.o.g., q_1 of them belong to cluster C_1, q_2 to cluster $C_2,\ldots,q_r$ to cluster C_r, $r \geqslant 2$. Then the contribution of x to the objective function is $q_1\delta_1+\cdots+q_r\delta_r$, where $\delta_h \sum_{k:0_k\in C_h} |x-x_k|$. Let $\delta_m=\min_{h=1,\ldots,r}\delta_h$. Re-assign all q replications of x to cluster C_m (if when doing so a cluster becomes empty this contradicts optimality). Then the contribution of x to the objective function becomes $q\delta_m \leqslant q_1\delta_1+\cdots+q_r\delta_r$. In this way, one obtains a new optimal partition where all replications of x belong to the same cluster. □

Note that not all optimal partitions need to have all entities at the same point in the same cluster, as shown by the next example.

Example 5. $N=4,\ x_1=0,\ x_2=x_3=1,\ x_4=2,\ M=2.$ $P_2^*=\{\{x_1,x_2,x_3\},\{x_4\}\}$ and $P_2^{*\prime}=\{\{x_1,x_2\},\{x_3,x_4\}\}$ are both optimal partitions with value $F^*=2$.

It follows from Proposition 4 that UECP may be viewed as clustering entities with positive integer multiplicities at distinct points on the line. In what follows, we will assume for simplicity of notation all multiplicities to be equal to 1. Modifications to address the general case are straightforward.

2. A polynomial algorithm

The nestedness property can be used to solve the univariate clique partitioning problem. Since one optimizes a function of N points on a line, dynamic programming appears to be the natural approach to exploit the total ordering of these points. Such an approach works well in the case of squared distances, for which the string property holds. This not being the case here one may consider performing computations in a bottom-up fashion in the encompassment tree. Then the optimal value for a complete subtree, i.e., a subtree induced by the set of all descendants of a given node C_k with $H(C_k)=[x_i,x_q]$ and encompassing $p-1$ cluster will be denoted by g_{iqp}. (The same notation g_{iqp} will be used to designate the total optimal value for several subtrees on disjoint consecutive subsets of points from $\{x_i,\ldots,x_q\}$.) This value and those of indices i, q and p are all the information regarding the subtree which is required for computation at the immediate predecessor node, C_h of C_k in the tree. Unfortunately, it is hard to derive an easily computable recursion for g_{iqp} as any subset of points of $\{x_{i+1},\ldots,x_{q-1}\}$ can be a potential child of C_k. One would then want to exploit the total ordering of points of C_k. To this effect consider an arbitrary point $x_\ell \in C'_k$ with $i<\ell<q$. The optimal value for the subtree rooted at C_k is the sum of

(i) optimal values for clusters immediately encompassed by C_k and left of x_ℓ;
(ii) optimal values for clusters immediately encompassed by C_k and right of x_ℓ;
(iii) distances between pairs of points of C'_k both to the left of or at x_ℓ;
(iv) distances between pairs of points of C_k both to the right of or at x_ℓ;
(v) distances between pairs of points on opposite sides of x_ℓ (see Fig. 2).

Contributions to the optimal value of points within $H(C_k)$ to the left of or at x_ℓ consists of (i), (iii) and part of (v). This last fact causes some difficulty when deriving a left to right recursion for the inner dissimilarity. The distances of (v) depend not only on the number but also on the actual position of the points of C_k to the right of v_ℓ. This difficulty is resolved by considering only the part of the distances of (v) on the left of and up to x_ℓ. One can compute this value knowing only the number of points of C_k to the right of x_ℓ.

More formally, we introduce the notion of "charge". Given two reals x, y (with $x \leqslant y$) the *relative distance* with respect to $[a,b]$ is defined as the length of $[x,y]\cap[a,b]$. Then the charge of a partition $P_L=\{C_1,C_2,\ldots,C_k\}$ with respect to $[a,b]$ is defined by

$$h_{[a,b]}P_L=\sum_{j=1,\ldots,L}\sum_{k,\ell\,|\,0_k,0_\ell\in C_j} d_{[a,b]}(x_k,x_\ell).$$

Clearly, the charge coincides with the sum of within cluster distances if $L=M$ and $[x_1,x_N]\subset[a,b]$.

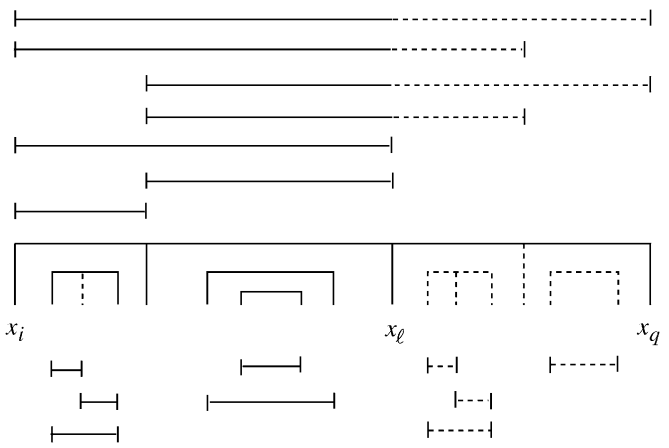

Fig. 2. Illustration of the concept of charge given in Eq. (1). (Broken segments not included in the charge.)

Consider an interval $[x_i, x_\ell]$ and a nested partition $P_M = \{C_1, C_2, \ldots, C_M\}$ such that:

(i) x_i is the first entity of C_k;
(ii) x_ℓ is the tth entity from left to right in C_k;
(iii) The set $[x_i, x_\ell]$ encompasses $p-1$ clusters of P_M different from C_k;
(iv) $|C_k| = s$ and hence $H(C_k) = [x_i, x_q]$ where $q = i + s - 1$.

In terms of charge, one has

$$h_{[x_i,x_\ell]}(P_M) = \sum_{j|C_j \subset [x_i,x_\ell]} d(C_j) + d(C_k \cap [x_i, x_\ell]) + (s-t) \sum_{v=i|0_v \in C_k}^{\ell-1} d_{v\ell}. \tag{1}$$

In words $h_{[x_i,x_\ell]}(P_N)$ is the sum of three terms (see Fig. 2): (i) within cluster distances for clusters with a range within $[x_i, x_\ell]$; (ii) distances between pairs of points of C_k within $[x_i, x_\ell]$; (iii) distances from points x_v within C_k to x_ℓ multiplied by the number $s-t$ of points of C_k outside of $[x_i, x_\ell]$. The algorithm recursively computes the minimum charge $f^{st}_{i\ell p}$ of a partition P_M satisfying properties (1)–(4).

The charge has two interesting features. First, one can compute the $g_{i\ell p}$ form the $f^{st}_{i\ell p}$. Indeed,

$$g_{i\ell p} = \mathrm{Min}\left(\min_s f^{ss}_{i\ell p} \min_{i \leqslant j \leqslant \ell, p_1 | p_1 + p_2 = p} (g_{ij p_1} + q_{j+1,\ell,p_2}) \right) \tag{2}$$

as $t = s$ implies that $\ell = q$,

$$d_{[x_i,x_\ell]}(x, y) = d_{xy} \quad \forall x, y \in C_k.$$

(The sum on the right-hand side of (2) corresponds to the case of several subtrees on connecting points of $\{x_i, \ldots, x_q\}$.) Second, and more important, there is a polynomial time computable recursion for the charge. Let x_j denote the point of C_k

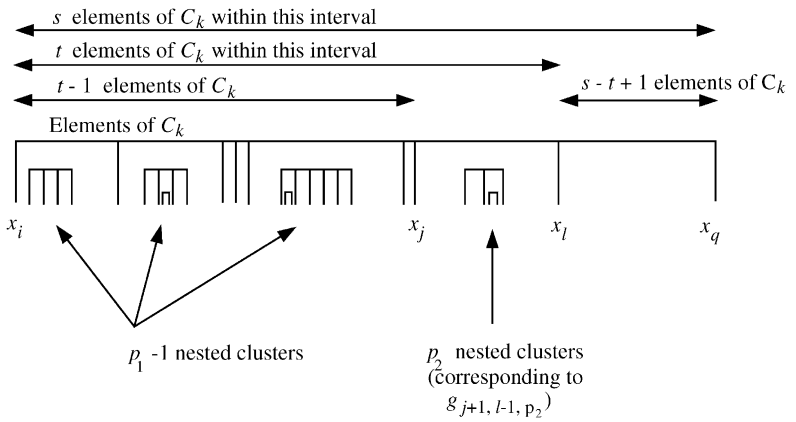

Fig. 3. Illustration of Eq. (3).

immediately preceding x_ℓ (see Fig. 3). Then, from (1) and Bellman's Optimality Principle,

$$f^{st}_{i\ell p} = \min_{\substack{i \leqslant j < \ell \\ p_1 + p_2 = p}} \{ f^{s,t-1}_{ijp_1} + g_{j+1,\ell-1,p_2} + (t-1)(s-t+1)(x_\ell - x_j) \}, \tag{3}$$

where p_1 and p_2 respectively, denote the number of clusters encompassed by $\{x_i, x_j\}$ and $\{x_j, x_\ell\}$ (see again Fig. 3).

The three terms on the right-hand side of (2) correspond to the charge of C_k and the $p_1 - 1$ clusters of P_M encompassed by C_k in $[x_i, x_j]$, the charge of the p_2 clusters of P_M in $[x_{j+1}, \ldots, x_{\ell-1}]$ (which is equal to their sum of interpoints distances) and the sum of relative distances $d_{[x_j,x_\ell]}(x, y)$ for pairs of points (x, y) of C_k ($t-1$ points of C_k are on the left of $]x_j, x_\ell[$ and $s-t-1$ on the right). The algorithm computes the charges $f^{st}_{i\ell p}$, in increasing order of $u = \ell - i$, and for all corresponding p, s and t. For all such i, ℓ and p after the charges have been found for all possible s and t the value $g_{i\ell p}$ is computed. Hence, when the left-hand side of (3) is computed, values needed on the right-hand side are available.

Rules of the algorithm are as follows:

Algorithm Univariate Euclidean Clique Partitioning (UECP)

(a) *Input*
Number N of entities, points $x_1 < x_2 < \cdots < x_N$, number of clusters M.

(b) *Initialization*
Set $g_{ii1} = f^{s1}_{ii1} = 0,\ i = 1, 2, \ldots, N,\ s = 1, 2, \ldots, N - i + 1$

(c) *General step*
For $u = 1$ to $N - 1$ do
For $i = 1$ to $N - u$ do
$\ell = i + u$

For $p=1$ to $\ell-i+1$ do
For $s=1$ to $N-i-p+1$ do
For $t=1$ to s do
Compute $f_{i\ell p}^{st}$ using formula (3)
(where $g_{j+1,\ell-1,p_2}=0$ if $j+1>\ell-1$) and set pointers to the largest j such that the minimum in (2) is attained and the corresponding largest p_2 (i.e., for that minimum value and j).
End For
End For
When all s and t have been examined, compute $g_{i\ell p}$ by formula (2) and (a) if the minimum is the first term on the right-hand side of (2): for the largest value of s such that this minimum is attained, list the indices of the corresponding cluster, as well as the g-values, enpoints and numbers of clusters that are children of C_k; (b) if the minimum is the second term on the right-hand side of (2): add pointers to the optimal g_{ijp_1}, g_{j+1,ℓ,p_2} and to their indices.
End For
End For
End For

(d) *Optimal Solution*

Using in top-down fashion the above information reconstruct the optimal partition $P_M^*=\{C_1^*,C_2^*,\ldots,C_M^*\}$.

Note that the tie-breaking rules on j, p_2 and s are arbitrary, but needed for the algorithms polynomial complexity to be guaranteed (In other words, finding all optimal nested partitions might not be polynomial).

Observe that algorithm UECP provides optimal partitions into $1,2,\ldots,M$ clusters. We next prove its correctness and evaluate its complexity.

Theorem 6. *Algorithm UECP provides an optimal clique partition into M clusters in* $O(N^5M^2)$ *time.*

Proof. To show that the algorithm is correct we must prove the validity of Eqs. (2) and (3). As both of them use results one from the other, we shall use simultaneous induction on $u=\ell-i$. For $u=0$ the initial values $g_{ii1}=f_{ii1}^{s1}=0$ given in the initialization step are correct. Suppose that there exists a smallest $u=\ell-i$ such that either Eq. (2) or Eq. (3) is incorrect. Consider first the latter case. By the minimality of u the values of $f_{ijp_1}^{s,t-1}$ and $g_{s+1,\ell-1,p_2}$ are correct. Hence, the left-hand side of Eq. (3) is correct by Bellman's optimality principle. Consider then the former case. From the induction hypothesis and correctness of Eq. (3) it follows that all values on the right-hand side of Eq. (2) are correct, and from the justification of this equation given above that the value on the left-hand side is correct as well. We thus have a contradiction: any

partition satisfying conditions (i)–(iv) and whose charge w.r.t. $[x_i, x_j]$ is different from the right-hand side of Eq. (3), or whose value for the complete subtree w.r.t. $[x_i, x_\ell]$ differs from the right-hand side of Eq. (2) is dominated.

Regarding complexity, the dominant step is the application of Eq. (3). Parameters i, ℓ, s and t take $O(N)$ values and p takes $O(M)$ values; the formula implies considering $O(N)$ values for j and $O(M)$ values for p_1, from which those of p_2 follow. So $O(N^5M^2)$ operations are required. □

3. The case $M = 2$

Complexity of the algorithm presented in Section 2 is rather high. A much more efficient algorithm may be obtained for the particular case where $M = 2$. There are 2 nested clusters. Let i and j denote the indices of the first and last entities of cluster C_1. Cluster C_2 consists of entities indexed by 1 to $i-1$ (if $i > 1$) and $j+1$ to N (if $j < N$). One cannot have both $i = 1$ and $j = N$ as $C_1, C_2 \neq \emptyset$. Let F_j^i denote the objective function value for that bipartition. The algorithm is based on efficient updating of F_j^i when i or j is increased one unit at a time. Consider first the case of fixed i. Let L_{j+1}^i denote the increase in sum of distances for C_1 when the entity indexed by $j+1$ is added to this cluster and R_{j+1}^i the corresponding decrease in sum of distances for C_2. Then

$$F_{j+1}^i = F_j^i + L_{j+1}^i - R_{j+1}^i, \tag{4}$$

where

$$L_{j+1}^i = \sum_{k=i}^{j} |x_k - x_{j+1}| = (j-i+1)x_{j+1} - \sum_{k=1}^{i} x_k \tag{5}$$

and

$$\begin{aligned} R_{j+1}^i &= \sum_{k=j+2}^{N} |x_{j+1} - x_k| + \sum_{k=1}^{i-1} |x_k - x_{j+1}| \\ &= (j+i-N)x_{j+1} + \sum_{k=j+2}^{N} x_k - \sum_{k=1}^{i-1} x_k. \end{aligned} \tag{6}$$

Proposition 7. *The update $L_{j+1}^i - R_{j+1}^i$ of F_j^i is an increasing function of j for all fixed $i < N/2 + 1$.*

Proof. From Eqs. (5) and (6)

$$L_{j+1}^i - R_{j+1}^i = (N - 2i + 2)x_{j+1} - \sum_{k=i}^{N} x_k + \sum_{k=1}^{i} x_k$$

which, when $i < N/2 + 1$, increases with j due to the ordering of the points. □

Corollary 8.

$$L^i_{j+1} - R^i_{j+1} - (L^i_j - R^i_j) = (N + 2 - 2i)|x_j - x_{j+1}|. \tag{8}$$

The next result shows one need not bother about j when $i \geqslant N/2 - 1$.

Proposition 9. *If $i \geqslant N/2 + 1$ then $j = N$ in all optimal solutions of univariate clique bipartitioning, i.e., the string property holds.*

Proof. Assume by contradiction $i \geqslant N/2 + 1$ and $j < N$. Then

$$|C_1| = j - i + 1 \leqslant N/2 < i \quad \text{and}$$
$$L^i_{j+1} - R^i_{j+1} \leqslant \sum_{k=i}^{j} |x_k - x_{j+1}| - \sum_{\ell=1}^{i} |x_\ell - x_{j+1}| < 0$$

as $|x_k - x_{j-1}| > |x_\ell - x_{j+1}|$ for $k = 1, \ldots, i - 1$ and $\ell = i, i + 1, \ldots, j$. But then the bipartition is not optimal. □

Let $j(i)$ be the smallest j such that $F^i_{j(i)} = \min_{j \geqslant i} F^i_j$.

Proposition 10. *Function $j(i)$ is nondecreasing in i.*

Proof. For C_1 one has

$$L^{i+1}_{j+1} - L^i_{j+1} = -|x_i - x_{j+1}|$$

and for C_2

$$R^{i+1}_{j+1} - R^i_{j+1} = |x_i - x_{j+1}|.$$

Hence,

$$L^{i+1}_{j+1} - R^{i+1}_{j+1} - (L^i_{j+1} - R^i_{j+1}) = -2|x_i - x_{j+1}| \tag{9}$$

which is negative. Reducing updates of F^i_j for a unit increase of j can only increase or keep constant the optimal value of j. □

We next consider updating F^i_j when i increases. Let D^i_j denote the decrease in sum of distances for C_1 when the i^{th} entity is transfered to C_2 and I^i_j the corresponding increase in the sum of distances for C_2. Then

$$F^{i+1}_j = F^i_j - D^{i+1}_j + I^{i+1}_j, \tag{10}$$

where

$$D^{i+1}_j = \sum_{k=i+1}^{j} |x_k - x_i|$$

and

$$I_j^{i+1} = \sum_{k=1}^{i-1} |x_k - x_i| + \sum_{k=j+1}^{N} |x_k - x_i|.$$

These quantities are updated as follows for a unit increase of i and fixed j (assuming without loss of generality that $j \geqslant i+1$; in the algorithm j will be increased first in the case $i = j$):

$$\begin{aligned} D_j^{i+1} - D_j^i &= \sum_{k=i}^{j} (|x_k - x_i| - |x_k - x_{i-1}|) \\ &= -\sum_{k=i}^{j} (|x_i - x_{i-1}|) = -(j-i+1)|x_i - x_{i-1}| \end{aligned}$$

and

$$\begin{aligned} I_j^{i+1} - I_j^i &= \sum_{k=1}^{i} (|x_k - x_i| - |x_k - x_{i-1}|) + \sum_{k=j+1}^{N} (|x_k - x_i| - |x_k - x_{i-1}|) \\ &= \sum_{k=1}^{i} |x_i - x_{i-1}| - \sum_{k=j+1}^{N} |x_i - x_{i-1}| = (i+j-N-1)|x_i - x_{i-1}|. \end{aligned}$$

Thus,

$$I_j^{i+1} - D_j^{i+1} - (I_j^i - D_j^i) = (2j - N)|x_i - x_{i-1}|. \tag{11}$$

For a unit increase of j and fixed i the updates are as follows:

$$D_{j+1}^{i+1} - D_j^{i+1} = |x_{j+1} - x_i| \tag{12}$$

and

$$I_{j+1}^{i+1} - I_j^{i+1} = -|x_j - x_i|. \tag{13}$$

The three propositions and the updating rules above are exploited in the following linear algorithm for univariate clique bipartitioning.

Algorithm Univariate Euclidean Clique Bipartitioning (UECB)

(a) *Input*

Number N of entities, points $x_1 < x_2 < \cdots < x_N$.

(b) *Initialization*

Set $i = 2$, $j = 2$,

$$F_2^2 = \sum_{k=3}^{N} |x_k - x_1| + \sum_{k=4}^{N} (N-k)(k-3)|x_k - x_{k-1}|$$

$$L_3^2 = |x_3 - x_2|$$

$$R_3^2 = \sum_{k=4}^{N} |x_3 - x_k| + |x_1 - x_3|$$

$$D_2^2 = 0$$

$$I_2^2 = |x_2 - x_1| + \sum_{k=3}^{N} |x_k - x_2|$$

$$F_{\text{opt}} = F_2^2, \quad i_{\text{opt}} = 2, \quad j_{\text{opt}} = 2.$$

(c) *General step: nested clusters*

While $i \leqslant \lfloor N/2 + 1 \rfloor$ and $j < N$

if $L_{j+1}^i - R_{j+1}^i < 0$ or $i = j$ then set $j = j + 1$

update F_j^i by (4)

update $L_{j+1}^i - R_{j+1}^i$ by (4)

update $D_j^{i+1} - I_j^{i+1}$ by (12) and (13)

if $L_{j+1}^i - R_{j+1}^i > 0$ and $i < j$ set $i = i + 1$

update F_j^i by (10)

update $L_{j+1}^i - R_{j+1}^i$ by (9)

update $I_j^i - D_j^i$ by (12) and (13)

if $F_j^i < F_{\text{opt}}$ set $F_{\text{opt}} = F_j^i$, $i_{\text{opt}} = i$, $j_{\text{opt}} = j$.

(d) *General step: clusters with string property*

While $i \leqslant N$, set $i = i + 1$

update F_N^i by (10)

update $I_N^i - D_N^i$ by (12) and (13)

if $F_N^i < F_{\text{opt}}$ set $F_{\text{opt}} = F_N^i$, $i_{\text{opt}} = i$, $j_{\text{opt}} = N$.

Theorem 11. *Algorithm UECB solves the univariate Euclidean bipartitioning problem in $\theta(N)$ time.*

Proof. Any bipartition is specified by the values of i and j. The algorithm considers all values of i and for each such value those successive values of j for which F_j^i decreases. That no values of j other than the last one lead to a better solution for that i exist follows from Propositions 7 and 10. The total number of updates of F_j^i and of updates of the updates is of $N - 1$ for i and $N - 1$ for j, i.e., $O(N)$. These updates are done in constant time following formulae (4)–(11). As reading the data takes $\Omega(N)$ time the algorithm is in $\theta(N)$. □

Due to the nestedness property, the case $M = 3$ can be solved in $O(N^3)$ time by considering all possible intervals $[x_i, x_j]$ for cluster C_1 removing the corresponding entities and applying algorithm UECB to the remaining ones. Similarly, an $O(N^5)$ algorithm may be obtained for the case $M = 4$.

As pointed out by an anonymous referee, Proposition 10 leads to another linear-time algorithm.

References

[1] R. Bellman, A note on cluster analysis and dynamic programming, Math. Biosci. 18 (1973) 311–312.

[2] E. Boros, P.L. Hammer, On clustering problems with connected optima in Euclidean spaces, Discrete Math. 75 (1989) 81–88.

[3] P. Brucker, On the complexity of clustering problems, in: M. Beckmann, H. Kunzi (Eds.), Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, Vol. 157, Springer, Heidelberg, 1978, pp. 45–54.
[4] M. Delattre, P. Hansen, Bicriterion cluster analysis, IEEE Trans. Pattern Analysis Machine Intelligence PAMI-2(4) (1980) 277–291.
[5] C. De Simone, M. Lucertini, S. Pallottino, B. Simeone, Fair dissections of spiders, worms and caterpillars, Networks 20 (1990) 323–344.
[6] J.C. Gower, G.J.S. Ross, Minimum spanning trees and single-linkage cluster analysis, Appl. Statist. 18 (1969) 54–64.
[7] P. Hansen, M. Delattre, Complete link cluster analysis by graph coloring, J. Amer. Statist. Assoc. 73 (1978) 397–403.
[8] P. Hansen, B. Jaumard, Computational methods in clustering from a mathematical programming viewpoint, in: H.H. Bock, W. Polasek (Eds.), Data Analysis and Information Systems, Studies in Classification, Data Analysis and Knowledge Organization, Springer, Heidelberg, 1996, pp. 24–40.
[9] P. Hansen, B. Jaumard, Cluster Analysis and Mathematical Programming, Math. Programming 79 (1997) 191–215.
[10] P. Hansen, B. Jaumard, E. Sanlaville, Partitioning problems in cluster analysis: a review of mathematical programming approaches, In: E. Diday et al. (Eds.), New Approaches in Classification and Data Analysis, Studies in Classification, Data Analysis and Knowledge Organization, Springer, Heidelberg, 1994, 228–240.
[11] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[12] F.K. Hwang, U.G. Rothblum, Y.-C. Yao, Localizing combinatorial properties of partitions, RUTCOR Research Report 2–94, Rutgers University, 1994.
[13] L. Kaufman, P.J. Rousseuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, NewYork, 1990.
[14] M. Lucertini, Y. Perl, B. Simeone, Most uniform path partitioning and its use in image processing, Discrete Appl. Math. 42 (1993) 227–256.
[15] M.R. Rao, Cluster analysis and mathematical programming, J. Amer. Statist. Assoc. 66 (1971) 622–626.
[16] M. Thorup, Randomized sorting in $O(n \log\log n)$ time and linear space using addition, shift and bitwise Boolean operations, Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA 1997, 352–359.
[17] W.J. Welch, Algorithmic complexity: three NP-Hard problems in computational statistics, J. Statist. Comput. Simulation 15 (1982) 17–25.