# Size-constrained 2-clustering in the plane with Manhattan distance

Alberto Bertoni, Massimiliano Goldwurm, Jianyi Lin, Linda Pini

Dipartimento di Informatica, Università degli Studi di Milano
Via Comelico 39/41, 20135 Milano – Italy,

**Abstract.** We present an algorithm for the 2-clustering problem with cluster size constraints in the plane assuming $\ell_1$-norm, that works in $O(n^3 \log n)$ time and $O(n)$ space. Such a procedure also solves a full version of the problem, computing the optimal solutions for all possible constraints on cluster sizes. The algorithm is based on a separation result concerning the clusters of any optimal solution of the problem and on an extended version of red-black trees to maintain a bipartition of a set of points in the plane.

**Keywords:** algorithms and data structures, clustering, cluster size constraints, Manhattan distance.

## 1 Introduction

Clustering is one of the most used techniques in statistical data analysis and machine learning [5], with a wide range of applications in many areas like pattern recognition, bioinformatics and image processing. Clustering consists in partitioning data into groups, called *clusters*, that are required to be homogeneous and well-separated according to a data similarity measure [9]. A natural representation of the data elements is by means of points in a $d$-dimensional space equipped with a suitable distance function that determines the similarity measure. We study this very usual case, called *distance clustering*.

A central problem in distance clustering is the well-known Minimum Sum-of-Squares Clustering (MSSC), also called Variance-based Clustering [2, 10]. The MSSC problem requires to find a $k$-partition $\{A_1, ..., A_k\}$ of a given set $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ of $n$ points that minimizes the weight

$$W(A_1, ..., A_k) = \sum_{i=1}^{k} \sum_{x \in A_i} ||x - C_{A_i}||_2^2$$

where $||\cdot||_2$ denotes the Euclidean norm and $C_{A_i}$ is the mean point of the cluster $A_i$, defined by

$$C_{A_i} = \operatorname*{argmin}_{\mu \in \mathbb{R}^d} \sum_{x \in A_i} ||\mu - x||_2^2 = \frac{1}{|A_i|} \sum_{x \in A_i} x.$$

MSSC is difficult in general; indeed for an arbitrary dimension $d$ (given in the instance) the problem is NP-hard even if the number of clusters is fixed to $k = 2$ [1, 8]; the same occurs if $k$ is arbitrary and the dimension is fixed to $d = 2$ [14]. However there's a well-known heuristic for finding an approximate solution of MSSC, called $k$-Means [13], which is known to be usually very fast, but can require exponential time in the worst case [17].

Moreover, a known variant of MSSC, called $k$-Medians Problem, is given by substituting the squared-euclidean norm $||\cdot||_2^2$ with the $\ell_1$-norm $||\cdot||_1$ [15].

In many applications, often people have some information on the clusters [3]: including this information into traditional clustering algorithms can increase the clustering performance. Problems that include such background information are called *constrained clustering* problems and are split into two classes. On the one hand, clustering problems with point-level constraints typically comprise a set of must-link constraints or cannot-link constraints [18], defining pairs of point that, respectively, must be or cannot be grouped in the same cluster. On the other hand, clustering problems with cluster-level constraints [6, 16] provides constraints concerning the size of the resulting clusters. For example, in [19] cluster size constraints are used for improving clustering accuracy; this approach, for instance, allows one to avoid extremely small or large clusters yielded by classical clustering techniques.

In this work we study the 2-clustering problem in the plane, with cluster size constraints, assuming $\ell_1$-norm. The relevance of the 2-clustering problems is also due to the wide spread of hierarchical clustering techniques [11, Sec.14.3.12], that repeatedly apply the 2-clustering as the key step. This leads to natural size-constrained versions of so-called divisive hierarchical clustering.

We recall that the 2-clustering problem with cluster size constraints has been studied in [12, 4], where it is shown that in dimension 1 the problem is solvable in polynomial time for every norm $\ell_p$ with integer $p \geq 1$, while there is some evidence that the same result does not hold for non-integer $p$. Moreover, it is also known that for arbitrary dimension $d$ the same problem is NP-hard even assuming equal sizes of the two clusters.

Here we prove that, assuming dimension 2 and $\ell_1$-norm, the 2-clustering problem with cluster size constraints can be solved in $O(n^3 \log n)$ time and $O(n)$ space. Our procedure actually solves a full version of the problem, computing the optimal solutions for all possible sizes of cluster. Clearly this yields the solution to the general unconstrained 2-clustering problem in the plane with $\ell_1$-norm.

The algorithm is based on a separation result stating that, in our hypotheses, the clusters of any optimal solution are separated by curves of some special forms, which can be analysed in a suitable linear order. The algorithm also make use of a particular extension of red-black trees to maintain a bipartition of a set of points in the plane.

We remark that in this work we propose an efficient method for obtaining a solution that is globally optimal, instead of a locally optimal solution as yielded by various heuristics [3, 6, 19].

## 2 Problem definition

In this section we introduce some basic notions of distance-based clustering problems on the plane assuming the Manhattan norm (also called $\ell_1$-norm), which is defined by $\|x\|_1 = |x_1| + |x_2|$ for any $x = (x_1, x_2) \in \mathbb{R}^2$.

Given a set $X \subset \mathbb{R}^2$ of $n$ points in the plane, a *k-partition* of $X$ is a family $\{A_1, A_2, ..., A_k\}$ of $k$ nonempty subsets of $X$ such that $\cup_{i=1}^{k} A_i = X$ and $A_i \cap A_j = \emptyset$, for $i \neq j$. Each $A_i$ is called *cluster*. The *centroid* $C_A$ of a cluster $A \subset X$ is

$$C_A = \operatorname*{argmin}_{\mu \in \mathbb{R}^2} \sum_{x \in A} \|x - \mu\|_1 = \operatorname*{argmin}_{\mu \in \mathbb{R}^2} \sum_{x \in A} (|x_1 - \mu_1| + |x_2 - \mu_2|)$$

Since the objective function in the minimization is not strictly convex, the centroid $C_A$ is not necessarily unique. Indeed it is well-known that the centroid $C_A$ is the component-wise median, i.e. the abscissa (ordinate) of the centroid is the median of the abscissae (ordinates) of the points in $A$. The *weight* $W(A)$ of a cluster $A$ is

$$W(A) = \sum_{x \in A} \|x - C_A\|_1$$

while the *weight* of a $k$-partition $\{A_1, A_2, ..., A_k\}$ (also called *k-clustering*) is

$$W(A_1, A_2, \cdots, A_k) = \sum_{1}^{k} W(A_i).$$

The classical *Clustering Problem* in the plane assuming the Manhattan norm is stated as follows.

**Definition 1 (Clustering Problem in $\mathbb{R}^2$ under $\ell_1$-norm).** *Given a point set $X \subset \mathbb{R}^2$ of cardinality $n$ and an integer $k$, $1 < k < n$, find a $k$-clustering $\{A_1, A_2, ..., A_k\}$ that minimizes the weight $W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i)$.*

Note that here $k$ is included in the instance of the problem. If $k$ is fixed the problem is called $k$-Clustering in $\mathbb{R}^2$. Our main contribution in this paper concerns the 2-Clustering in $\mathbb{R}^2$ under $\ell_1$-norm.

We are interested in a version of clustering problem where the cluster sizes are constrained. Formally, the problem can be stated as follows:

**Definition 2 (Size Constrained Clustering Problem in $\mathbb{R}^2$ under $\ell_1$-norm).** *Given a point set $X \subset \mathbb{R}^2$ of cardinality $n$, an integer $k > 1$ and $k$ positive integers $m_1, m_2, ..., m_k$ such that $\sum_1^k m_i = n$, find a $k$-clustering $\{A_1, A_2, ..., A_k\}$ with $|A_i| = m_i$ for $i = 1, ..., k$, that minimizes the weight $W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i)$.*

We denote this problem by SCC-2 (under $\ell_1$-norm). We stress that in the SCC-2 problem the integers $n$, $k$, $m_1, \ldots, m_k$ are part of the instance. On the contrary, if $k$ is fixed and does not belong to the instance, the problem is denoted by $k$-SCC-2.

In the following sections we present an algorithm for the 2-Clustering problem in $\mathbb{R}^2$ under $\ell_1$-norm, which solves at the same time the *full* version of the 2-SCC-2 problem, i.e. that yields the solutions to all 2-SCC-2 problems for every value of $m_1 \in \{1, 2, \ldots, \lfloor n/2 \rfloor\}$ and the same point set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^2$ in input. The procedures works in $O(n^3 \log n)$ time and is based on a separation result concerning the optimal solutions of 2-Clustering in the plane, presented in Section 3, and on an augmented version of red-black trees used to maintain bipartitions of a set of real points, described in Section 4.

## 3 Separation results

In this section we present a separation result concerning the optimal solution of 2-SCC-2 problem under $\ell_1$-norm. It turns out that the clusters of the optimal solutions of this problem are separated by plane curves of 8 different types, we will call $C_i$ and $S_i$, $i = 1, 2, 3, 4$, respectively. Similar results are obtained in [4] for higher dimensional spaces and $\ell_p$-norms, with $p > 1$.

First, we show that in an optimal 2-clustering the swapping of two elements in different clusters does not decrease the solution value.

**Lemma 1 (Swapping Lemma).** *Let $\{A, B\}$ be an optimal solution of the 2-SCC-2 problem. Then, for any $a \in A$ and $b \in B$, it holds*

$$||a - C_A||_1 + ||b - C_B||_1 \leq ||a - C_B||_1 + ||b - C_A||_1 .$$

*Proof.* For the sake of simplicity, we omit the subscript 1 in denoting norm. By contradiction, let us assume that for some $a \in A$ and $b \in B$

$$||a - C_A|| + ||b - C_B|| > ||a - C_B|| + ||b - C_A|| .$$

Then, the weight of the optimal solution is

$$\begin{aligned} W(A, B) &= \sum_{x \in A} ||x - C_A|| + \sum_{x \in B} ||x - C_B|| = \\ &> \sum_{x \in A \setminus \{a\}} ||x - C_A|| + ||b - C_A|| + \sum_{x \in B \setminus \{b\}} ||x - C_B|| + ||a - C_B|| = \\ &= \sum_{x \in A \setminus \{a\} \cup \{b\}} ||x - C_A|| + \sum_{x \in B \setminus \{b\} \cup \{a\}} ||x - C_B|| . \end{aligned}$$

Now, it is clear that the 2-clustering $\{A', B'\}$, with $A' = A \setminus \{a\} \cup \{b\}$ and $B' = B \setminus \{b\} \cup \{a\}$, is a feasible solution since $|A'| = |A|$ and $|B'| = |B|$. Furthermore, by definition of centroid it holds:

$$\sum_{x \in A'} ||x - C_A|| + \sum_{x \in B'} ||x - C_B|| \geq \sum_{x \in A'} ||x - C_{A'}|| + \sum_{x \in B'} ||x - C_{B'}|| = W(A', B')$$

and hence $W(A, B) > W(A', B')$. However, this is in contradiction since $\{A, B\}$ is an optimal solution. □

We are now able to obtain the general form of the equation for the curves separating the clusters in an optimal 2-clustering.

**Theorem 1 (Separation Result).** *In an optimal solution $\{A, B\}$ of the 2-SCC-2 problem, the clusters $A$ and $B$ are separated by a curve of equation*

$$||z - \alpha||_1 - ||z - \beta||_1 = g \tag{1}$$

*where the variable $z$ ranges over $\mathbb{R}^2$, while $\alpha, \beta \in \mathbb{R}^2$ and $g \in \mathbb{R}$ are suitable constants.*

*Proof.* Let $C_A$ and $C_B$ be the centroids of $A$ and $B$ respectively. Then, by Lemma 1 the inequality

$$||a - C_A||_1 - ||a - C_B||_1 \leq ||b - C_A||_1 - ||b - C_B||_1$$

is satisfied for all $a \in A$ and $b \in B$, and hence it also holds by taking the maximum over $a$'s on the left and the minimum over $b$'s on the right:

$$c_A := \max_{a \in A}\{||a - C_A||_1 - ||a - C_B||_1\} \leq \min_{b \in B}\{||b - C_A||_1 - ||b - C_B||_1\} =: c_B$$

Therefore, choosing $g \in [c_A, c_B]$, all the points of $A$ and $B$ are contained in the region defined respectively by $||z - C_A||_1 - ||z - C_B||_1 \leq g$ and $||z - C_A||_1 - ||z - C_B||_1 \geq g$. The common boundary of the two regions has equation $||z - C_A||_1 - ||z - C_B||_1 = g$, and thus the proof is concluded by setting $\alpha = C_A$ and $\beta = C_B$. □

*Remark 1.* It is noteworthy to observe that both the Swapping Lemma and the Separation Result can be extended to the case of $\ell_p$-norm with any $p \geq 1$ and dimension $d \geq 2$, provided the norm is raised to the $p$-th power [12].

Setting the symbols $z = (x, y)$, $\alpha = (c, d)$, $\beta = (e, f)$, equation (1) becomes

$$|x - c| - |x - e| + |y - d| - |y - f| = g$$

It is clear that it always represents a connected curve on the plane, which may have some sharp corner points, i.e. points where one or both partial derivatives do not exist. A rather standard study of this equation, conducted by considering all possible values of $c, d, e, f, g$, leads to derive 8 types of separating curves, we denote by symbols $C_1, C_2, C_3, C_4$ and $S_1, S_2, S_3, S_4$, according to their shape. The proof is here omitted for lack of space. Such 8 types of curves are depicted in Figures 1–4.

## 4 Bipartition red-black trees

In this section we describe a natural data structure to maintain a bipartition of a set of real numbers, with possible multiple values, able to test membership, to execute insertion and deletion, and to perform two further operations: selecting

the $i$-th smallest element in either of the two clusters of the bipartition, for any integer $i$, and moving an element from one cluster to the other.

Let $X$ be a finite multiset of real numbers, i.e. a finite set $X$ of reals where any value may occur several times, represented as a finite ordered sequence

$$X = (x_1, x_2, \ldots, x_n), x_i \in \mathbb{R} \text{ for every } i, \text{ and } x_1 \leq x_2 \leq \cdots \leq x_n.$$

Moreover, consider a bipartition $\{A, B\}$ of $X$, i.e. two non-empty subsets $A \subseteq X$, $B \subseteq X$ such that $A \cap B = \emptyset$ and $A \cup B = X$. In the following $A$ and $B$ are called clusters. Operations Member, Insert and Delete can be defined in order to specify the cluster the elements belong to. More precisely, for every $z \in \mathbb{R}$, Member$(z) = (j_1, j_2)$, where $j_1$ (resp. $j_2$) is the number of $x_i$'s in $A$ (resp. in $B$) such that $x_i = z$, Insert$(z, A)$ adds a new element $z$ to $A$, Insert$(z, B)$ does the same to $B$, while Delete$(z, A)$ and Delete$(z, B)$ execute the delete operations.

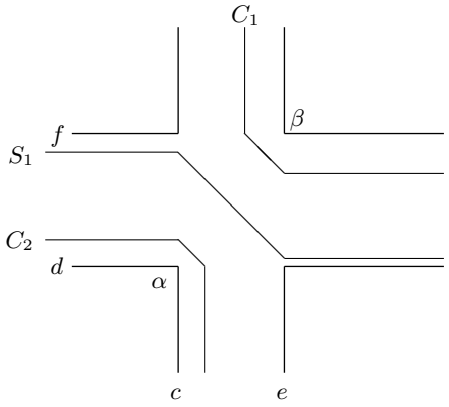

Figure 1

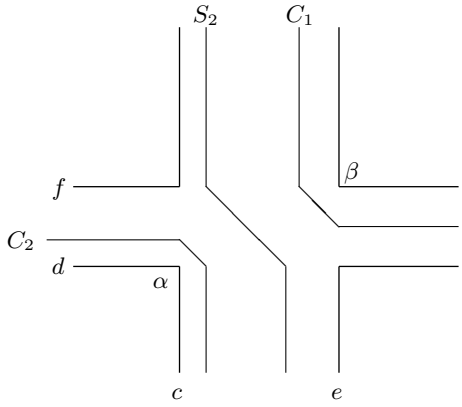

Figure 2

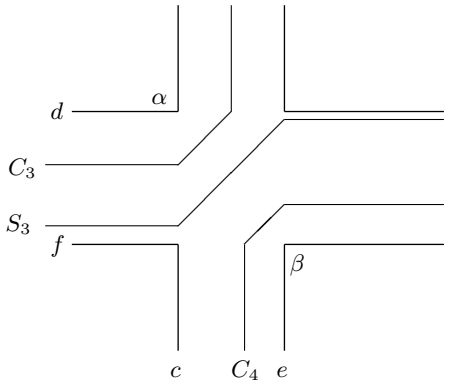

Figure 3

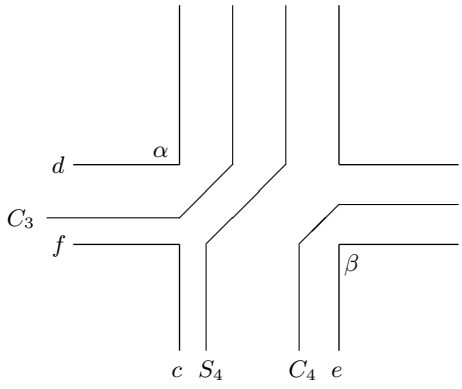

Figure 4

Moreover, for every $i \in \{1, 2, \ldots, n\}$, we define

$$\text{Select}_A(i) = \begin{cases} a \text{ if } i \leq |A| \text{ and } a \text{ is the } i\text{-th smallest element of } A \\ \perp \text{ if } i > |A| \end{cases}$$
$$\text{Select}_B(i) = \begin{cases} b \text{ if } i \leq |B| \text{ and } b \text{ is the } i\text{-th smallest element of } B \\ \perp \text{ if } i > |B| \end{cases}.$$

Finally, for any $z \in \mathbb{R}$, if $A$ contains an element $x_i = z$ then $\text{Move}_{AB}(z)$ deletes a value $z$ from $A$ and adds it to $B$, otherwise $\text{Move}_{AB}(z) = \perp$. The operation $\text{Move}_{BA}(z)$ is defined symmetrically.

A natural data structure, able to executes the above operations in time logarithmic with respect to $n = |X|$, is an augmented version of the well-known red-black trees [7]. Formally, we define a *Bipartition Red-Black Tree* for a bipartition $\{A, B\}$ of $X$ as a binary tree $T$ such that: *i*) all internal nodes have both the left and the right son, *ii*) every node is red or black, *iii*) the root and the leaves are black, *iv*) if a node is red its sons are black, and *v*) for every node $v$ all simple paths from $v$ to any leaf have the same number of black nodes. Further, the distinct elements of $X$ (called keys) are assigned to the internal nodes in a bijective correspondence, respecting the standard rule of binary search trees, and the leaves are empty (usually represented by "nil"). Moreover, $T$ satisfies the following two conditions:

1. every internal node $v$ contains a triple $(z, m_A(v), m_B(v))$, where $z$ is the key assigned to $v$, and $m_A(v)$ (respectively, $m_B(v)$) is the number of elements $x_i$ belonging to $A$ (resp. $B$) such that $x_i = z$;
2. every internal node $v$ contains the pair of integers $(c_A(v), c_B(v))$, where $c_A(v)$ (resp. $c_B(v)$) is the number of elements of $X$ belonging to $A$ (resp. $B$) whose value is assigned to a vertex in the subtree rooted at $v$.

For each internal node $v$, we denote by $key(v)$, $left(v)$ and $right(v)$ the key contained in $v$, the left son and the right son of $v$, respectively. Clearly, $v$ can be implemented as a record including the values $m_A(v)$, $m_B(v))$, $c_A(v)$, $c_B(v)$.

Procedures Member, Insert and Delete can be defined as in the standard red-blacks trees [7] with obvious changes. Clearly, Insert and Delete have to update the values $m_A(v)$, $m_B(v))$, $c_A(v)$, $c_B(v)$, for all nodes $v$ on the path from the root to the vertex resulting from the binary search.

Operations Select and Move can be executed as follows. To compute $\text{Select}_A(i)$ one checks whether $i \leq c_A(r)$, where $r$ is the root of $T$: in the affirmative case a standard searching procedure (we avoid to define for lack of space) is called, otherwise the value $\perp$ is returned. $\text{Select}_B(i)$ is defined in a similar way.

To compute $\text{Move}_{AB}(z)$ one first determines the node $v$ containing $z$ by calling a rather usual procedure, then the commands

$$m_A(v) = m_A(v) - 1 \quad \text{and} \quad m_B(v) = m_B(v) + 1$$

are executed. Procedure $\text{Move}_{BA}(z)$ is defined symmetrically.

It is clear that all previous procedures can be executed in $O(\log n)$ time.

## 5 Updating the weight of clusters

Our main result, presented in the next section, is based on a procedure that updates a current bipartition by moving one point from a cluster to the other. Here we want to show how to update the centroids and the weights of the clusters of a current bipartition after executing such a moving operation.

To this end we first deal with the problem of updating centroid and weight of a set of real numbers subject to insert and delete operations. The results will be easily extended to a set of points in $\mathbb{R}^2$.

Let $A$ be a cluster in $\mathbb{R}$, i.e. a finite sequence of ordered real numbers, $A = \{a_1, a_2, \ldots, a_k\}$, where $a_1 \leq a_2 \leq \cdots \leq a_k$. Assuming $\ell_1$-norm it is well-known that the centroid of $A$ is the median of the set, that is the value

$$M(A) = \begin{cases} a_{\frac{k+1}{2}} & \text{if } k \text{ is odd} \\ \frac{1}{2}\left(a_{\frac{k}{2}} + a_{\frac{k}{2}+1}\right) & \text{if } k \text{ is even} \end{cases}$$

Thus, if $A$ is a cluster of a bipartition (of a finite multiset $X \subset \mathbb{R}$ of $n$ elements), maintained by a Bipartition Red-Black Tree, then $M(A)$ can be computed in $O(\log n)$ time by calling procedure $\text{Select}_A(j)$ for suitable $j$'s.

Now, let us recall the definition of the left and right part of the weight of $A$; given $m = \frac{k+1}{2}$, we have

$$L(A) := \sum_{1 \leq i < m} a_i \ , \qquad R(A) := \sum_{m < i \leq k} a_i$$

It is not difficult to see that the weight of $A$, given by $W(A) = \sum_{i=1}^{k} |a_i - M(A)|$, can be computed as the difference between $R(A)$ and $L(A)$.

**Proposition 1.** *For every ordered sequence of real numbers $A = \{a_1, a_2, \ldots, a_k\}$, where $a_1 \leq a_2 \leq \cdots \leq a_k$, we have*

$$W(A) = R(A) - L(A)$$

*Proof.* If $k$ is even then $m = \frac{k+1}{2}$ is not integer and hence

$$\begin{aligned} W(A) = \sum_{i=1}^{k} |a_i - M(A)| &= \sum_{1 \leq i < m} (M(A) - a_i) + \sum_{m < i \leq k} (a_i - M(A)) \\ &= -L(A) + R(A) \end{aligned}$$

If $k$ is odd the reasoning is almost the same. □

As a consequence, in order to maintain $W(A)$ it is sufficient to update $L(A)$ and $R(A)$. This can be done by a straightforward procedure that implements, as a chain of if-then-else instructions, the rules for computing the left and right part of the weight of a current cluster upon insertion and deletion. For lack of space here we omit their cumbersome formal description. These rules together with

Proposition 1, define procedures UpdateW-ins$(A, x)$ and UpdateW-del$(A, a_j)$, which update the weight of a current cluster $A \subset \mathbb{R}$, respectively upon insertion and deletion of an element of value $x \in \mathbb{R}$.

They can be executed by using a Bipartition Red-Black tree with the corresponding Select operation, introduced in the previous section. Note that, for updating both $L(A)$ and $R(A)$, the Select procedure is necessary to find in the tree the $i$-th value of $A$ when required. As a consequence also UpdateW-ins$(A, x)$ and UpdateW-del$(A, a_j)$ can be executed in logarithmic time.

## 6 Main algorithm

In this section we apply the previous results, in particular the separation property of Section 3, to define an algorithm for the 2-SCC-2 problem in the full version, i.e. for all possible sizes of cluster. Thus the same algorithm also yields a solution of the general 2-Clustering problem in the plane.

By our Theorem 1, the optimal solution of a 2-SCC-2 problem for a set $X \subset \mathbb{R}^2$ of $n$ points and a cluster size $m$, consists of a bipartition $(A, B)$, with $|A| = m$ and $|B| = n - m$, where $A$ and $B$ are separated by a curve of the form $C_i$ or $S_i$, $i = 1, 2, 3, 4$.

Note that each curve of type $C_i$ can be obtained from a curve of any other form $C_j$, $j \neq i$, by means of a simple rotation of the plane. Similarly each curve of type $S_i$ can be obtained from another curve of any different type $S_j$ by means of a rotation and/or a reflection through a line.

For these reasons we treat in detail only the case of bipartitions separated by a curve of type $C_1$. The cases of curves of type $C_2, C_3$ and $C_4$ are solved by the same algorithm through a rotation of the input points. The same algorithm can be easily adapted to the cases of separating curves of type $S_i$, $i = 1, 2, 3, 4$.

Let $X = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$ be a set of $n$ points and let us set $p_i = (x_i, y_i)$, for any $i = 1, 2 \ldots, n$. Without loss of generality we may assume that all $p_i$'s lie in the first quadrant, i.e. $x_i > 0$ and $y_i > 0$ for all $i$.

We define a procedure that searches an optimal bipartition of $X$ whose clusters are separated by a curve of type $C_1$. The procedure first sorts $X$ according with 3 parameters: the two coordinates and their sum, i.e. the values $x_i$'s, $y_i$'s and $x_i + y_i$'s, respectively. Then the elements of $X$ are processed by three for-loops, each nested into the other, corresponding to the three parameters considered above. The algorithm evaluates all possible bipartitions of $X$ whose clusters are separated by curves of type $C_1$ and, for every admissible cluster size, only the bipartition with minimum weight is maintained. These bipartitions are considered in a linear order and, at each step, a new bipartition of $X$ is obtained from the current one $\{A, B\}$ by swapping a point from cluster $B$ to cluster $A$.

Such a current bipartition $\{A, B\}$ is maintained by two bipartition red-black trees: one for the 2-clustering $\{A_x, B_x\}$ of the set $\{x_i \mid p_i = (x_i, y_i) \in X\}$, where $A_x = \{x_i \mid p_i \in A\}$ and $B_x = \{x_i \mid p_i \in B\}$, and the other for the 2-clustering $(A_y, B_y)$ of $\{y_i \mid p_i = (x_i, y_i) \in X\}$, where $A_y = \{y_i \mid p_i \in A\}$ and $B_y = \{y_i \mid p_i \in B\}$.

An Update subroutine can be defined which, for an input $(A, B, p_i)$, where $\{A, B\}$ is the current bipartition of $X$ and $p_i \in B$, computes the new bipartition moving $p_i$ from $B$ to $A$ and returns the weights of its clusters. The subroutine applies the procedures described in Sections 4 and 5 and it is defined by the following scheme:

```
Procedure Update(A, B, p_i)
begin
    Move_{B_x A_x}(x_i)
    Move_{B_y A_y}(y_i)
    w(A_x) := UpdateW-ins(A_x, x_i)
    w(B_x) := UpdateW-del(B_x, x_i)
    w(A_y) := UpdateW-ins(A_y, y_i)
    w(B_y) := UpdateW-del(B_y, y_i)
    return (w(A_x) + w(A_y), w(B_x) + w(B_y))
end
```

By the discussion presented in Sections 4 and 5 also this subroutine only requires $O(\log n)$ time.

Thus, the overall algorithm is described in detail by the procedure given below, where three bipartitions are actually maintained, denoted by $(A, B)$, $(A', B')$ and $(A'', B'')$, respectively. Each of them need two Bipartition Red-Black trees, one for the abscissae and the other for the ordinates.

Moreover, instruction $\mathrm{Sort}_x$ (resp., $\mathrm{Sort}_y$ and $\mathrm{Sort}_{x+y}$) yields an ordered list of the indices of the points in $X$, sorted in non-decreasing order with respect to the values $x_i$ (resp., $y_i$ and $x_i + y_i$).

Taking into account the previous discussion, it is easy to see that the algorithm works in $O(n^3 \log n)$ time; moreover, it has a space complexity $O(n)$ since each Bipartition Red-Black tree only requires linear space.

```
Procedure Full-biclustering(p_1, p_2, ..., p_n)
begin
    (i_1, i_2, ..., i_n) := Sort_x(1, 2, ..., n)
    (j_1, j_2, ..., j_n) := Sort_y(1, 2, ..., n)
    (k_1, k_2, ..., k_n) := Sort_{x+y}(1, 2, ..., n)
    A := ∅ ; w(A) := 0
    B := {p_1, p_2, ..., p_n}
    compute the weight w(B) of cluster B
    i_0 := 0 ; j_0 := 0 ; x_0 := 0 ; y_0 := 0
    for r = 0, 1, ..., n do
        A' := A ; B' := B ; w(A') := w(A) ; w(B') := w(B)
        for s = 0, 1, ..., n do
            A'' := A' ; B'' := B' ; w(A'') := w(A') ; w(B'') := w(B')
            for k = k_1, k_2, ..., k_n do
                if x_k ≥ x_{i_r} ∧ y_k ≥ y_{j_s} then
                    (w(A''), w(B'')) = Update(A'', B'', p_k)
```

$h := \min(|A''|, |B''|)$
$z := w(A'') + w(B'')$
`if` $W[h] > z$ `then` $\begin{cases} W[h] := z \\ \Pi[h] := (i_r, j_s, k) \end{cases}$
`if` $s \neq n \wedge x_{j_{s+1}} \geq x_{i_r}$ `then`
$(w(A'), w(B')) = \text{Update}(A', B', p_{j_{s+1}})$
$h := \min(|A'|, |B'|)$
$z := w(A') + w(B')$
`if` $W[h] > z$ `then` $\begin{cases} W[h] := z \\ \Pi[h] := (i_r, j_{s+1}, 0) \end{cases}$
`if` $r \neq n$ `then`
$(w(A), w(B)) = \text{Update}(A, B, p_{i_{r+1}})$
$h := \min(|A|, |B|)$
$z := w(A) + w(B)$
`if` $W[h] > z$ `then` $\begin{cases} W[h] := z \\ \Pi[h] := (i_{r+1}, 0, 0) \end{cases}$
`return` $(W, \Pi)$
`end`

## 7 Conclusions

In this paper we have shown a $O(n^3 \log n)$ time algorithm for the size-constrained 2-Clustering problem in the plane with $\ell_1$-norm. It is clear that our results strictly depend on those conditions and it is not evident (at least at the present time) whether they hold under different hypotheses or for more general problems. Moreover, being our algorithm exact and working in polynomial time, we do not think it is necessary to produce here, in our hypotheses (2-clustering in $\mathbb{R}^2$ with $\ell_1$-norm), an experimental comparison with traditional heuristics like $k$-Means, which instead yields an approximate solution, tackles a more general problem with larger number of clusters, and works in exponential time in the worst case.

However, we think that the present contribution could be the starting point for a further analysis, still based on the separation result introduced in [4], of more general clustering problems, with dimension greater than 2, larger number of clusters, different norms and other constraints. It is clear that such a research direction should also include an experimental activity that plans a comparison with classical heuristics.

We also remark we have already obtained recently some results under other hypotheses. In particular, in a companion paper we present more efficient algorithms for solving exactly the same problem with Euclidean norm, based on rather different techniques of computational geometry. More precisely, it turns out that the 2-Clustering problem in $\mathbb{R}^2$ under Euclidean norm, with size constraint $k$, can be solved by an exact algorithm in $O(n\sqrt[3]{k}\log^2 n)$ time, while the full version of the same problem (i.e. for all $k = 1, 2, \ldots, \lfloor n/2 \rfloor$) is solvable in $O(n^2 \log n)$ time.

## References


1. D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75:245–249, 2009.
2. D. Aloise and P. Hansen. On the complexity of minimum sum-of-squares clustering. Technical report, Les Cahiers du GERAD, 2007.
3. S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman and Hall/CRC, 2008.
4. A. Bertoni, M. Goldwurm, J. Lin, and F. Saccà. Size Constrained Distance Clustering: Separation Properties and Some Complexity Results. *Fundamenta Informaticae*, 115(1):125–139, 2012.
5. C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
6. P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained K-Means Clustering. Technical Report MSR-TR-2000-65, Miscrosoft Research Publication, May 2000.
7. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.
8. S. Dasgupta. The hardness of $k$-means clustering. Technical Report CS2007-0890, Dept. Computer Sc. and Eng., Univ. of California, San Diego, 2007.
9. W. D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.
10. S. Hasegawa, H. Imai, M. Inaba, and N. Katoh. Efficient algorithms for variance-based $k$-clustering. In *Proceedings of Pacific Graphics '93*, pages 75–89, 1993.
11. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2nd edition, 2009.
12. J. Lin. *Exact algorithms for size constrained clustering*. PhD Thesis, Università degli Studi di Milano. Ledizioni Publishing, Milano, 2013.
13. J. B. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Structures*, pages 281–297, 1967.
14. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar $k$-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.
15. K. Sabo. Center-based l1-clustering method. *International Journal of Applied Mathematics and Computer Science*, 24(1):7–225, 2014.
16. A. Tung, J. Han, L. Lakshmanan, and R. Ng. Constraint-Based Clustering in Large Databases. In J. Van den Bussche and V. Vianu, editors, *Database Theory ICDT 2001*, volume 1973 of *LNCS*, pages 405–419. Springer Berlin/Heidelberg, 2001.
17. A. Vattani. K-means requires exponentially many iterations even in the plane. In *Proceedings of the 25th Symposium on Computational Geometry (SoCG)*, 2009.
18. K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. of the 17th Intl. Conf. on Machine Learning*, pages 1103–1110, 2000.
19. S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.