On the Complexity of Clustering with Relaxed Size Constraints

Massimiliano Goldwurm(2), Jianyi Lin(1), Francesco Saccà (1)

(1) Dipartimento di Informatica, (2) Dipartimento di Matematica
Università degli Studi di Milano, 20100 Milano – Italy

Abstract. We study the computational complexity of the problem of computing an optimal clustering $\{A_1, A_2, ..., A_k\}$ of a set of points assuming that every cluster size $|A_i|$ belongs to a given set M of positive integers. We present a polynomial time algorithm for solving the problem in dimension 1, i.e. when the points are simply rational values, for an arbitrary set M of size constraints, which extends to the ℓ_1-norm an analogous procedure known for the ℓ_2-norm. Moreover, we prove that in the Euclidean plane, i.e. assuming dimension 2 and ℓ_2-norm, the problem is NP-hard even with size constraints set reduced to $M = \{2, 3\}$.

Keywords: geometric clustering problems; cluster size constraints; computational complexity; constrained k-Means

1 Introduction

In the area of unsupervised machine learning and statistical data analysis the clustering methods play an important role with applications in pattern recognition, bioinformatics, signal and image processing, medical diagnostics. Clustering consists in grouping a set of objects into subsets, called clusters, that are maximally homogeneous [5,8]. Partitional or hard clustering requires the subsets to be disjoint and non-empty, and in the usual geometric setting the similarity between objects is measured by distance between points representing the objects [15].

A classical clustering problem is the so-called Euclidean Minimum-Sum-of-Squares [1], Variance-based [10] or k-Means clustering problem: given a finite point set $X \subset \mathbb{R}^d$, find a k-partition $\{A_1, ..., A_k\}$ of X minimizing the sum of weights $W(A_1, ..., A_k) = \sum_i W(A_i) = \sum_i \sum_{x \in A_i} \|x - \mu(A_i)\|^2$ of all clusters, where $\mu(A_i)$ is the sample mean of A_i and $\| \cdot \|$ is the Euclidean norm. This partitional clustering problem is difficult: when d is part of the instance the problem is NP-hard even if the number of clusters is fixed to $k = 2$ [1]; the same occurs for arbitrary k with fixed dimension $d = 2$ [7,16]. Nonetheless, a well-known heuristic for this problem is Lloyd's algorithm [14], also named k-Means Algorithm, which is not guaranteed to converge to the global optimum. This algorithm is usually very fast, but may require exponential time in the worst case [22].

Often one has some a-priori information on the clusters, that can be incorporated into traditional clustering techniques to increase the clustering performance [2]. Problems that include background information are so-called constrained clustering and can be divided into two classes based on the constraints: instance-level constraints typically define pairs of elements that must be (must-link) or cannot be (cannot-link) in the same cluster [25], and cluster-level constraints prescribe characteristics of each cluster, such as cluster diameter or cluster size [6,21]. In [26] cluster size constraints are used for improving clustering accuracy, for instance allowing one to avoid extremely small or large clusters in standard cluster analysis. In the *size constrained clustering* (SCC) problem, assuming an ℓ_p-norm (we suppose $p \in \mathbb{N}_+$ throughout this work), typically one is given a finite set $X \subset \mathbb{R}^d$ of n points and k positive integers $m_1, ..., m_k$ such that $\sum_i m_i = n$, and searches for a partition $\{A_1, ..., A_k\}$ of X, with $|A_1| = m_1, ..., |A_k| = m_k$, that minimizes the objective function $W(A_1, ..., A_k) = \sum_{i=1}^{k} \sum_{x \in A_i} \|x - c_i\|_p^p$, where each $c_i = \operatorname{argmin}_{c \in \mathbb{R}^d} \sum_{x \in A_i} \|x - c\|_p^p$ is the ℓ_p-centroid of A_i.

For arbitrary $k \in \mathbb{N}$, this problem is NP-hard also in dimension $d = 1$, for any (fixed) ℓ_p-norm, $p \geq 1$; the same negative result holds for arbitrary $d \in \mathbb{N}$ when the number of clusters is fixed to $k = 2$, for every ℓ_p-norm with $p > 1$ [3]. On the contrary, in the case $d = 2 = k$ the problem is solvable in $O(n^2 \log n)$ time assuming Manhattan norm (ℓ_1) and in $O(n \sqrt[3]{m} \log^2 n)$ time with Euclidean norm (ℓ_2) [13], where m is the size of one of the two clusters.

In this work we study a *relaxed version* of the size constrained clustering problem, where the size of each cluster belongs to given set M of integers. We show that in dimension $d = 1$, for an arbitrary (finite) $M \subset \mathbb{N}$, assuming the Manhattan norm, the solution can be obtained in $O(n(ks + n))$ time, where k is the number of clusters and s is the cardinality of M. This extends an analogous algorithm [4] proposed for the Euclidean norm and applied to computational biology problems as a method for identification of promoter regions in genomic sequences. Note instead that, in dimension 1, the SCC problem is NP-hard [3]. On the contrary, in dimension $d = 2$, we prove that even fixing $M = \{2, 3\}$ the problem is NP-hard with Euclidean norm.

2 Problem definition

In this section we define the problem and fix our notation. Given a positive integer d, for every real $p \geq 1$ and every point $a = (a_1, ..., a_d) \in \mathbb{R}^d$, we denote by $\|a\|_p$ the ℓ_p-norm of a, i.e. $\|a\|_p = (\sum_1^d |a_i|^p)^{1/p}$. Clearly, $\|a\|_2$ and $\|a\|_1$ are the Euclidean and the Manhattan (or Taxicab) norm of a, respectively.

Given a finite set $X \subset \mathbb{R}^d$, a *cluster* of X is a non-empty subset $A \subset X$, while a *clustering* is a partition $\{A_1, ..., A_k\}$ of X in k clusters for some k. Assuming the ℓ_p norm, the *centroid* and the *weight* of a cluster A are the values $C_A \in \mathbb{R}^d$ and $W_p(A) \in \mathbb{R}_+$ defined, respectively, by

$$C_A = \operatorname*{argmin}_{c \in \mathbb{R}^d} \sum_{a \in A} \|a - c\|_p^p, \qquad W_p(A) = \sum_{a \in A} \|a - C_A\|_p^p$$

The *weight* of a clustering $\{A_1, ..., A_k\}$ is $W_p(A_1, ..., A_k) = \sum_1^k W_p(A_i)$. We recall that, in case of ℓ_2-norm, the weight of a cluster A can be computed by relation

$$W_2(A) = \frac{1}{|A|} \sum_{(*)} \|a - b\|_2^2 \tag{1}$$

where the sum is extended to all unordered pairs $\{a, b\}$ of distinct elements in A. Moreover, given a set $\mathcal{M} \subset \mathbb{N}$, any clustering $\{A_1, ..., A_k\}$ such that $|A_i| \in \mathcal{M}$ for every $i = 1, \ldots, k$, is called $\mathcal{M}$-*clustering*.

RSC-d Problem (with ℓ_p-norm): Relaxed Size Constrained Clustering in $\mathbb{R}^d$
Given a set $X \subset \mathbb{Q}^d$ of n points, an integer k such that $1 < k < n$ and a finite set $\mathcal{M}$ of positive integers, find an $\mathcal{M}$-clustering $\{A_1, ..., A_k\}$ of X that minimizes $W_p(A_1, ..., A_k)$. [1]

When $\mathcal{M}$ is not included in the instance, but fixed in advance, we call the problem $\mathcal{M}$-**RSC-d** (with ℓ_p-norm). In this work we study these problems in dimension $d = 1, 2$ assuming ℓ_1 and ℓ_2-norm.

3 Dynamic programming for RSC on the line

In this section we describe a polynomial-time algorithm for RSC-1 that works assuming either ℓ_1 or ℓ_2-norm. This procedure is based on a dynamic programming technique, in the style of [19], based on the so-called String Property [24,3]. A simplified version of the procedure in the case of ℓ_2-norm is also presented in [4] and applied to problems of computational biology.

Consider an instance $(X, k, \mathcal{M})$ of RSC-1, where $X = (x_1, x_2 ..., x_n)$ is a sorted sequence of rational numbers, $k \in \{1, \ldots, n-1\}$ and $|\mathcal{M}| = s \leq n$. For any $1 \leq i \leq j \leq n$, let $X[i, j]$ be the subsequence $(x_i, x_{i+1}, ..., x_j)$. For a given $p \in \{1, 2\}$, we define the $n \times n$ matrix $U = [U(i, j)]_{i,j=1,...,n}$ by setting $U(i, j) = W_p(X[i, j]) = \sum_{t=i}^{j} |x_t - C_{X[i,j]}|^p$ if $j - i + 1 \in \mathcal{M}$ and $U(i, j) = \infty$ otherwise, that is the weight of cluster $X[i, j]$ when it has admissible size.

Lemma 1. *Given $p \in \{1, 2\}$, for every instance $(X, k, \mathcal{M})$ of RSC-1 with $|X| = n$, matrix U can be computed in $O(n^2)$ time.*

Proof. First, assume $p = 2$. In this case it is easy to check that the weight of any cluster A is $W_2(A) = \sum_{a \in A} a^2 - \frac{1}{|A|}(\sum_{a \in A} a)^2$. Denoting $Q(i) := \sum_{j=1}^{i} x_j^2$ and $S(i) := \sum_{j=1}^{i} x_j$, the finite entries of matrix U reduce to

$$U(i, j) = Q(j) - Q(i-1) - \frac{1}{j - i + 1}(S(j) - S(i-1))^2. \tag{2}$$

The sequences Q and S can be computed in linear time, and thus the computation of (2) requires constant time for each i, j. Hence, matrix U can be computed in $O(n^2)$ time in case $p = 2$.

[1] If X does not admit a $\mathcal{M}$-clustering then symbol $\perp$ is returned.

When $p = 1$, the weight $W_1(X[i,j])$ is the sum of the distances between elements and median of $X[i,j]$. Denote $m := (i+j)/2$ and for any cluster $X[i,j]$ set the left and right sums $L(i,j) := \sum_{i \le h < m} x_h$ and $R(i,j) := \sum_{m < h \le j} x_h$. It can be checked ([3, Prop. 10]) that $W_1(X[i,j]) = R(i,j) - L(i,j)$. Since $X[i,j]$ is sorted, it can be seen that, for $i < j$,

$$L(i,j) = L(i,j-1) \text{ if } m \in \mathbb{N}, \quad L(i,j) = L(i,j-1) + x_{\lfloor m \rfloor} \text{ otherwise,} \tag{3}$$

$$R(i,j) = R(i,j-1) - x_m + x_j \text{ if } m \in \mathbb{N}, \quad V = R(i,j-1) + x_j \text{ otherwise} \tag{4}$$

and $L(i,i) = R(i,i) = 0$. By means of these recursive formulae the quantities $L(i,j), R(i,j), W_1(X[i,j])$, for all $i \le j$, can be computed in $O(n^2)$ time, and hence the same holds for determining matrix U when $p = 1$. □

Now, for every $h \in \{1,\ldots,k\}$ and every $j \in \{1,\ldots,n\}$, let $Z(h,j)$ be the weight of a solution of RSC-1 for the instance $(X[1,j], h, \mathcal{M})$ in case $h \le j$, while $Z(h,j) = \infty$ if $h > j$. These values can be derived from U.

Proposition 2. *The following properties hold:*
i) $Z(1,j) = U(1,j)$ *for all* $j = 1,...,n$*;*
ii) $Z(h,j) = \min_{m \in \mathcal{M}} (Z(h-1,j-m) + U(j-m+1,j))$ *for all* $h = 2,..,k; j = 1,...,n$.

Proof. Case *i*) is obvious. Since $Z(h,j)$ is the weight of the optimal solution for $(X[1,j], h, \mathcal{M})$, the corresponding solution $\{A_1,...,A_h\}$ satisfies the String Property, i.e. each cluster A_i consists of consecutive points of X [24,3].

Then, its right-most cluster A_h has size $|A_h| = m \in \mathcal{M}$ and weight $W_p(A_h) = W_p(X[j-m+1,j]) = U(j-m+1,j)$.

The other clusters $A_1,...,A_{h-1}$ form a feasible clustering of RSC-1 for the instance $(X[1,j-m], h-1, \mathcal{M})$, which has minimum weight $W_p(A_1,...,A_{h-1}) = \sum_1^{h-1} W_p(A_i) = Z(h-1,j-m)$, otherwise it is easy to check that also $\{A_1,...,A_h\}$ would not be an optimal solution for $(X[1,j], h, \mathcal{M})$.

As a consequence, $Z(h,j) = Z(h-1,j-m) + U(j-m+1,j)$ for some $m \in \mathcal{M}$, and since $Z(h,j)$ has to take the minimum value, property *ii*) is proved. □

Relying on the previous proposition we can design an algorithm for RSC-1.

Theorem 3. *For any* $p \in \{1,2\}$*, RSC-1 with* ℓ_p*-norm can be solved in* $O(n(ks+n))$ *time and* $O(n^2)$ *space.*

Proof. By Lemma 1 we first compute matrix U in $O(n^2)$ time. Then, by means of Proposition 2, matrix $Z = [Z(h,j)]_{h=1,\ldots,k; j=1,\ldots,n}$ can be computed row by row. Each entry requires at most $s = |\mathcal{M}|$ sums and comparisons. The computation is described by the following scheme, where we store in $\ell_{h,j}$ the size of the last cluster of the optimal solution for $(X[1,j], h, \mathcal{M})$, for each pair of indices h, j.

```
begin
  Z := {∞}^{k×n}
```

```
for j = 1,...,n do { Z(1,j) := U(1,j)
                   { if U(1,j) ≠ ∞ then ℓ_{1,j} := j
for h = 2,...,k do
    for j = h,...,n do
        m̂ := argmin_{m∈M}{Z(h−1, j−m) + U(j−m+1, j)}
        if m̂ is well-defined then
            Z(h,j) := Z(h−1, j−m̂) + U(j−m̂+1, j)
            ℓ_{h,j} := m̂
end
```

Clearly, if $\ell_{k,n}$ is not defined then the symbol $\perp$ is returned since no admissible clustering for $(X, k, \mathcal{M})$ exists. Otherwise, the solution of the problem can be obtained by the following procedure:

```
begin
  j := n
                           { t_h := ℓ_{h,j}
  for h = k, k−1,...,1 do  { A_h := X[j − t_h + 1, j]
                           { j := j − t_h
  output {A_1, A_2, ..., A_k}
end
```

The overall time required to compute matrices U and Z is $O(n(ks+n))$. The space necessary to maintain all tables is $O(n^2)$ since $k < n$. □

It is worth noting that the analogous problem, where the size of each cluster is fixed by the instance, is NP-hard even in dimension $d = 1$ for every ℓ_p-norm [3]. This shows that the form of the size constraints for clustering problems is relevant for the existence of polynomial time algorithms.

4 NP-hardness of RSC in the Euclidean Plane

In this section we show that, assuming ℓ_2-norm, the $\{2,3\}$-RSC-2 problem is NP-hard, and therefore RSC-2 also is NP-hard. To this end we introduce a decision version of the problem and describe a polynomial-time reduction from Planar 3-SAT.

Decision $\{2,3\}$-RSC-2 Problem
Given a point set $X = \{p_1, ..., p_n\} \subset \mathbb{Q}^2$, an integer k, $1 < k \leq n/2$, and a rational value $\lambda > 0$ (threshold), decide whether there exists a $\{2,3\}$-clustering $\{A_1, ..., A_k\}$ of X, consisting of k clusters, such that $W_2(A_1, ..., A_k) \leq \lambda$.

Recall that a 3-CNF formula Φ is a boolean formula given by the conjunction of clauses each of which has 3 literals. If V and C are, respectively, the set of variables and the set of clauses of Φ, the *graph of Φ* is defined as the undirected bipartite graph G_Φ such that $V \cup C$ is the family of nodes and $E = \{\{v, c\} : v \in V, c \in C,$ and either v or $\bar{v}$ appears in $c\}$ is the set of edges. A formula Φ is said to be *planar* if G_Φ is planar. The Planar 3-SAT problem consists in deciding whether a planar 3-CNF formula Φ is satisfiable.

It is known that Planar 3-SAT is strongly NP-complete [12]. It is also proved that it suffices to consider formulae whose associated graph can be embedded in $\mathbb{R}^2$, with variables arranged on a straight line, and with clauses arranged above and below the straight line [11]. Moreover, the edges between variables and clauses can be drawn in a rectilinear fashion [17].

We also recall that a *box-orthogonal drawing* of a graph G is a planar embedding of G on an integer grid where each vertex is mapped into a (possibly degenerate) rectangle and each edge becomes a path of horizontal or vertical segments of the grid. Rectangles are disjoint and paths do not intersect. Any planar graph of n nodes admits a box-orthogonal drawing computable in $O(n)$ time that uses a $a \times b$ grid, where $a + b \leq 2n$ [9, Th. 3].

Our goal is to show that Planar 3-SAT is reducible in polynomial time to Decision $\{2, 3\}$-RSC-2. The proof is obtained by adapting the reduction from Planar 3-SAT to an unconstrained version of the k-means problem in the plane, presented in [16]. Here, the main difference is that in our construction we determine directly the rational coordinates of the points given by the reduction, avoiding the approximation of irrational values. Moreover, our reduction does not yield multiple copies of the same point in the plane.

To describe the reduction we show how an arbitrary planar 3-CNF formula Φ, can be associated with an instance (X, k, λ) of the Decision $\{2, 3\}$-RSC-2, computable in polynomial time w.r.t. $|\Phi|$, such that Φ is satisfiable if and only if X admits a partition into k clusters of cardinality 2 or 3, having a total weight at most λ. The definition of such a reduction is split in several phases: the first one computes an embedding of graph G_Φ into a planar integer grid; the others determine the rational coordinates of points in X, and the values k and λ.

The general idea is to build an embedding of G_Φ by representing each clause by a point in the grid, and associating each variable with a cycle on the grid that connects all points of clauses containing the variable. Clearly, these cycles do not overlap, and each clause-point is touched exactly by 3 cycles. Now, the points of X are placed along every cycle, so that there are only 2 optimal $\{2\}$-clusterings for the points of each cycle, which may be associated to the truth assignments of the variable. The satisfiability of each clause will correspond to the possibility of clustering the clause-point with the nearest pair in one of the optimal $\{2\}$-clusterings associated to a variable occurring in the clause.

1) Embedding of G_Φ into a planar grid

The first phase is described by the following steps, illustrated in Figure 1.

Step 0. Compute the box-orthogonal drawing D of G_Φ as stated above. We can map any variable into a (non-empty) rectangle and any clause into a vertex of the grid. Moreover, the base of all rectangles can be put on the same horizontal straight line, and the vertices representing clauses above or below such a line.

Step 1. Expand the previous drawing by a factor of 2 and call D_1 the new drawing. This doubles all distances between vertices in D.

Step 2. Shift D_1 half unit upward and rightward and let D_2 be the new drawing. Now, each clause corresponds to a point in the centre of a unit square of the grid, and each path from a rectangle (variable) to a point (clause) crosses just in the middle some unit sides of the grid.
Step 3. Expand all rectangles by half grid unit in all four vertical and horizontal directions, and replace any point (clause) of D_2 by a unit square centred at the same location, erasing the overlapping portion (half unit long) of paths. We call D_3 the new drawing. Now, all rectangles have sides of odd length and no path in D_3 starts from a vertex of a rectangle.
Step 4. Replace every path from a rectangle (variable) to a unit square (clause) by a *strip* of unit width on the grid that cover the same path, erasing the boundary portion of rectangle overlapping the strip. The resulting drawing is called D_4. Now every variable v corresponds to a (sort of) *cycle* on the grid that includes both the residual rectangle representing v and all strips towards the unit squares (clauses) where v occurs, together with one side for each touched square.

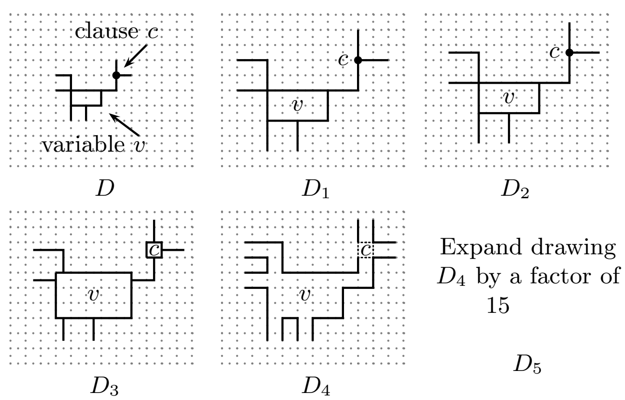

Fig. 1. Main steps of the graph transformations used in the reduction.

Step 5. Expand the previous drawing by a factor of 15. We call D_5 the new drawing. Thus, each clause is now associated with a square on the grid having side of length 15, while the strips described in Step 4 are formed by parallel segments at distance 15 to each other. Moreover, in the following we call *borders* the straight-line segments forming the cycles associated with the variables.

2) Definition of point set X

Let $V = \{v_1, \ldots, v_n\}$ and $C = \{c_1, \ldots, c_m\}$ be, respectively, the set of variables and the set of clauses of Φ. First, for every $c_j \in C$, X contains a point $z_j \in \mathbb{Q}^2$ located near the centre of the square associated with c_j. The exact position of each z_j is defined by Fig. 2, where the cycles are represented by dashed lines and the sides of the square are removed for sake of simplicity.

Moreover, for every variable $v_i \in V$, X contains a circuit Γ_i of $2L_i$ consecutive points $\{x_{i1}, x_{i2}, ..., x_{i(2L_i)}\}$, for a suitable integer L_i. With few exceptions (as in Fig. 2), all $x_{i\ell}$'s lie inside the cycle of drawing D_5 associated with v_i and inside the square associated with the clauses where v_i occurs. The idea is to put almost all points at distance 2 from the borders, setting at distance 5 from each other most consecutive points x_{it}, $x_{i(t+1)}$, as well as points $x_{i(2L_i)}$ and x_{i1}. Hence

$$X = \{z_j \mid j = 1, 2, \ldots, m\} \cup \{x_{i\ell} \mid i = 1, 2, \ldots, n,\ \ell = 1, 2, \ldots, 2L_i\} \qquad (5)$$

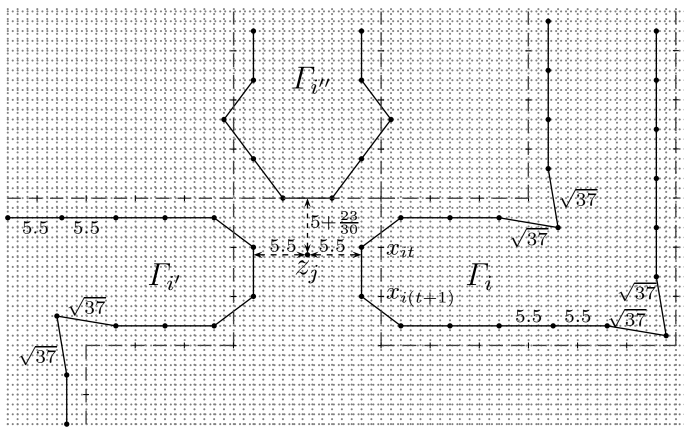

Fig. 2. Points of 3 circuits in the neighbourhood of a clause-point z_j. Edges with length different from 5 are indicated.

The precise position of points $x_{i\ell}$'s is illustrated in Figs. 2 and 3 and is formally defined by conditions a), b), c) given below. Such a position depends on the angles, inside the cycle associated with v_i, formed between two incident borders. Every angle has measure either $\pi/2$ or $3\pi/2$; in the first case we say the angle is *convex*, in the second case we say it is *concave* (e.g., in Fig. 3, angle β is convex, while α is concave).

a) Near every convex (resp. concave) angle three consecutive points of Γ_i are placed as shown by angles $\beta, \delta, \varepsilon, \zeta, \eta$ (resp. $\alpha, \gamma, \theta, \iota$) in Fig. 3. Note that the second point of the triple always lies on the bisector.

b) Between any two consecutive angles, the other points of Γ_i are put on a straight-line at distance 2 from the border, so that consecutive points are set at distance 5 from each other, with the exception of two segments of length 4.5 (respectively, 5.5) if both angles are concave (resp., convex). As examples, see in Fig. 3 points between angles β and γ, ι and θ, δ and ε.

c) If v_i, $v_{i'}$, $v_{i''}$ are the variables occurring in a clause c_j, then near the square of size 15×15 associated with c_j, points of Γ_i, $\Gamma_{i'}$, $\Gamma_{i''}$ are set as defined in Fig. 2. Note that here, all pair of consecutive points are at distance 5 from each other with two exceptions:
- triple of points close to angles are located according to condition a);
- before convex angles, points are located to form two consecutive segments of length 5.5.

3) Weight of clusters

Note that all pairs of consecutive points in any $\mathit{\Gamma}_i$ form a segment having one of the following lengths: 4.5, 5, 5.5, $\sqrt{37}$. The weight of the corresponding clusters is easily obtained from Eq. (1): 10.125, 12.5, 15.125, 18.5.

Moreover, every set $\mathit{\Gamma}_i$ admits only two $\{2\}$-clusterings of minimum weight, consisting of pairs of consecutive points, given by

$$\pi_1(i) = \{\{x_{iu}, x_{i(u+1)}\} \mid u = 1, 3, 5, \ldots, 2L_i - 1\} \qquad \text{and}$$
$$\pi_2(i) = \{\{x_{iu}, x_{i(u+1)}\} \mid u = 2, 4, 6, \ldots, 2L_i - 2\} \cup \{\{x_{i(2L_i)}, x_{i1}\}\}$$

For simplicity, hereafter we call *segment* (respectively, *triangle*) a cluster of cardinality 2 (resp., 3).

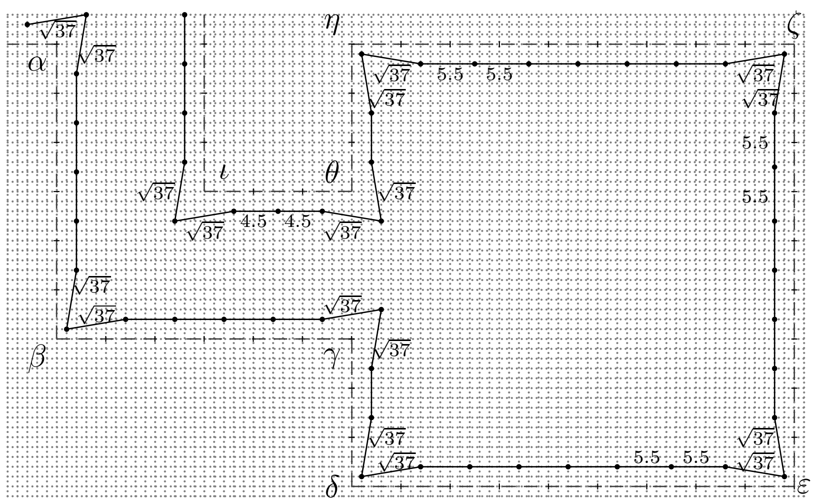

Fig. 3. Points of a circuit $\mathit{\Gamma}_i$ inside the corresponding rectangle. Segments with length different from 5 are indicated. Note that angles $\alpha, \gamma, \theta, \iota$ are concave, while $\beta, \delta, \varepsilon, \zeta, \eta$ are convex.

Now, consider a clause c_j containing a variable v_i (as a positive or negative literal) and let x_{it}, $x_{i(t+1)}$ be the pair of points of $\mathit{\Gamma}_i$ nearest to point z_j. We say that z_j *touches* the segment $\{x_{it}, x_{i(t+1)}\}$. Clearly every z_j touches three segments, one for each variable appearing in c_j. Note from Fig. 2, that the distance between z_j and a touched segment is either 5.5 or $5 + \frac{23}{30}$. Then, using Eq. (1), by elementary computation we can determine the weight of any triangle formed by each z_j with its touched segments, as well as the weight of every triangle of consecutive points in any $\mathit{\Gamma}_i$. Such a direct computation proves the following property.

Lemma 4. *If point z_j touches a segment $\{x_{it}, x_{i(t+1)}\}$ then the weight of triangle $\{z_j, x_{it}, x_{i(t+1)}\}$ is given by $w = \frac{23402}{675}$, which satisfies $34.66 < w < 34.67$. Moreover, every triangle composed by points of $\mathit{\Gamma}_i$ has weight greater than w.*

4) Parity condition

By a suitable choice of the first point x_{i1}, and possibly by adding new points to $\mathit{\Gamma}_i$ (as explained below), we can assume that the following parity condition holds: in any $\mathit{\Gamma}_i$, every segment touched by a point z_j belongs to either $\pi_1(i)$ or $\pi_2(i)$ according to whether v_i or $\overline{v_i}$ appears in c_j, respectively. In order to guarantee this property, slight changes to points of $\mathit{\Gamma}_i$ near the square including z_j may be necessary, which are illustrated in Figure 4. This change add two new points (one before and one after the touched segment), and determines 4 more segments of length 4.5, two of which are to be included into $\pi_1(i)$, the others into $\pi_2(i)$. In order to apply this transformation the circuit must contain a rectilinear portion of length at least 30, either horizontal or vertical, as shown in Fig. 4 (left). We may always assume this is satisfied by requiring one more expansion of the initial drawing by a factor of 2 (executing Step 1 twice in the embedding phase).

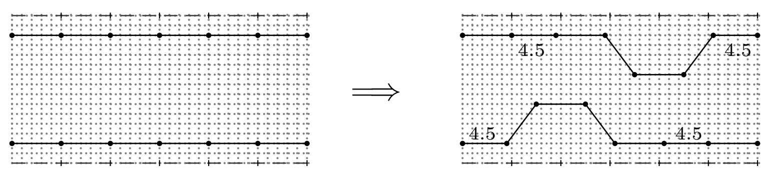

Fig. 4. (Left) 30×15 horizontal strip preserving parity. (Right) 30×15 horizontal strip for changing parity. The vertical case in analogous.

5) Definition of k and λ

They are given by equalities $k = \sum_1^n L_i$ and

$$\lambda = \frac{5^2}{2}(k-h) + wm + \frac{1}{2}\left[18.5 \cdot s_{\sqrt{37}} + \left(10 + \frac{1}{8}\right) \cdot s_{4.5} + \left(15 + \frac{1}{8}\right) \cdot s_{5.5}\right],$$

where w is defined as in Lemma 4, s_u is the total number of segments of length u in all $\mathit{\Gamma}_i$'s for $u \in \{\sqrt{37}, 4.5, 5.5\}$, and $h = m + \frac{1}{2}(s_{\sqrt{37}} + s_{4.5} + s_{5.5})$.

It is easy to see that every $\{2,3\}$-clustering π of X into k clusters must contain exactly m triangles. Indeed, if n_T and n_S denote, respectively, the number of triangles and the number of segments of π, then $|X| = 2n_S + 3n_T = 2k + m$ and $n_S + n_T = k$, which yields $n_T = m$ and $n_S = k - m$. Recall that all triangles in X have weight at least w. Moreover, by construction, π may include at most $s_u/2$ many segments of length u for each $u \in \{\sqrt{37}, 4.5, 5.5\}$ and the remaining $k - h$ cannot have length smaller than 5. This implies $W_2(\pi) \geq \lambda$.

Now, to complete the reduction we verify that $\mathit{\Phi}$ is satisfiable if and only if there exists a $\{2,3\}$-clustering of X of weight at most λ, consisting of k clusters. Suppose $\mathit{\Phi}$ is satisfiable and consider a satisfying assignment. For each variable v_i, choose clustering $\pi_2(i)$ or $\pi_1(i)$ according whether its value is 0 or 1, respectively. Since the assignment makes all clauses true, each point z_j can be clustered together with the touched segment in $\mathit{\Gamma}_i$, for a variable v_i satisfying clause c_j. By the parity condition, such a touched segment belongs to the chosen clustering (either $\pi_2(i)$ or $\pi_1(i)$). Thus, we obtain m triangles of weight w. The other points in each Γ_i can be clustered as in $\pi_2(i)$ or $\pi_1(i)$ according to the previous choice. This yields a $\{2,3\}$-clustering of X of weight λ having k clusters.

Vice-versa, if there exists a $\{2,3\}$-clustering π of X with k clusters and weight λ, then such a clustering must contain m triangles of weight w. The only way to obtain these triangles is to include each point z_j into a touched segment $\{x_{it}, x_{i(t+1)}\}$. By the parity condition this defines an assignment of values to all variables that makes true each clause of Φ.

Theorem 5. *Assuming ℓ_2-norm, the $\{2,3\}$-RSC-2 problem is strongly NP-hard and it does not admit an FPTAS unless $P = NP$. As a consequence, the same holds in general for RSC-2 problem.*

Proof. The NP-hardness follows from the discussion above. The problem is also strongly NP-hard since the value of all integers in instances (X, k, λ) obtained by the reduction is polynomially bounded w.r.t. $n = |X|$. Moreover, the objective function to minimize is polynomially bounded with respect to the unary size of the instance, and hence, by a classical result [23, Sec. 8.3], the same problem does not admit an FPTAS unless $P = NP$. □

5 Conclusions

In this work, we have studied the clustering problem with relaxed size constraints in dimension 1 and 2 (RSC-1 and RSC-2). First, we have shown a polynomial-time algorithm for RSC-1 in the case of ℓ_1 and ℓ_2-norm. A natural question is whether similar algorithm exists also for ℓ_p-norm with integer $p > 2$. We recall that the clustering in dimension 1 is motivated by bioinformatics applications as illustrated in [4].

Our second result states that $\mathcal{M}$-RSC-2 problem is strongly NP-hard when $\mathcal{M} = \{2,3\}$. Note that with $\mathcal{M} = \{2\}$ the problem reduces to finding a perfect matching of minimum cost in a weighted complete graph, and hence it is solvable in $O(n^3)$ time (even in arbitrary dimension) assuming any ℓ_p-norm, by using classical algorithms [18]. The same occurs when $\mathcal{M} = \{1,2\}$ since this is reducible to finding the minimum cost matching of given cardinality in a weighted graph, which is known to be solvable in polynomial time (see for instance [20, sec. 3.1.1]). Hence, a natural problem is to determine the sets $\mathcal{M}$ for which the $\mathcal{M}$-RSC-2 problem is NP-hard.

Finally, we conjecture that $\{2,3\}$-RSC-2 remains NP-hard also in case of ℓ_1-norm, by a suitable extension of the proof above.

Acknowledgments We thank an anonymous referee for his/her useful comments on other problems related to clustering with relaxed size constraints.

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. S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman and Hall/CRC, 2008.
3. 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.
4. A. Bertoni, M. Rè, F. Saccà, and G. Valentini. Identification of promoter regions in genomic sequences by 1-dimensional constraint clustering. In *Neural Nets WIRN11*, pages 162–169, 2011.
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. S. Dasgupta. The hardness of k-means clustering. Technical Report CS2007-0890, Dpt. of Computer Science and Engineering, Univ. of California, San Diego, 2007.
8. W. D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.
9. U. Fößmeier, G. Kant, and M. Kaufmann. 2-Visibility drawings of planar graphs. In *Graph Drawing (Proc. GD '96)*, volume 1190 of *LNCS*, pages 155–168, 1997.
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. D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.
12. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
13. J. Lin, A. Bertoni, and M. Goldwurm. Exact algorithms for size constrained 2-clustering in the plane. *Theoretical Computer Science*, 2015. doi:10.1016/j.tcs.2015.10.005. In Press.
14. S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
15. J. B. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. on Math. Struct.*, pages 281–297, 1967.
16. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.
17. W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2), 2008.
18. C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.
19. M. R. Rao. Cluster Analysis and Mathematical Programming. *Journal of the American Statistical Association*, 66(335):622–626, 1971.
20. R. Stephan. Cardinality constrained combinatorial optimization: Complexity and polyhedra. *Discrete Optimization*, 7(3):99 – 113, 2010.
21. A. Tung, J. Han, L. Lakshmanan, and R. Ng. Constraint-based clustering in large databases. In *Database Theory ICDT 2001*, volume 1973 of *LNCS*, pages 405–419.
22. A. Vattani. k-means Requires Exponentially Many Iterations Even in the Plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011.
23. V. Vazirani. *Approximation Algorithms*. Springer, 2001.
24. H. Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64(326):506 – 519, 1969.
25. K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. 17th Intl. Conf. on Machine Learning*, pages 1103–1110, 2000.
26. S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.