On the Complexity of Clustering with Relaxed Size Constraints in Fixed Dimension[1]

Massimiliano Goldwurm[(1)], Jianyi Lin[(2)], Francesco Saccà[(3)]

(1) Dipartimento di Matematica, (3) Dipartimento di Informatica
Università degli Studi di Milano, 20100 Milano – Italy
(2) Department of Applied Mathematics and Sciences
Khalifa University, Abu Dhabi - United Arab Emirates

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 Euclidean norm. Moreover, we prove that in dimension 2, assuming Euclidean norm, the problem is (strongly) NP-hard with size constraints $M = \{2, 4\}$. This result is extended also to the size constraints $M = \{2, 3\}$ both in the case of Euclidean and ℓ_1-norm.

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 with respect to a suitable criterion for evaluating the similarity of objects [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 [17].

A classical clustering problem is the so-called Euclidean Minimum-Sum-of-Squares [1], Variance-based [11] 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

[1]Appeared in revised form in *Theoretical Computer Science*, vol. 717, pp. 37-46, 2018. A preliminary version of this work was presented at the 11th AAIM Conference [10].

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 (also called ℓ_2-norm). In most cases such a 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, 7]; the same occurs for arbitrary k with fixed dimension $d = 2$ [18]. Nonetheless, a well-known heuristic for this problem is Lloyd's algorithm [15], 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 [25].

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 [28], and cluster-level constraints prescribe characteristics of each cluster, such as cluster diameter or cluster size [6, 24]. In [29] 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 with integer $p \geq 1$, 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 $k \in \mathbb{N}$, the SCC problem is NP-hard also in dimension $d = 1$, for any (fixed) ℓ_p-norm, $p \geq 1$ [3]. 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 SCC 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) [14], where m is the size of one of the two clusters.

In this work we study a *relaxed version* of the SCC problem, where the size of each cluster belongs to a given set M of integers, rather than being fixed by the instance of the problem. We show that in dimension $d = 1$, assuming the ℓ_1-norm, for an arbitrary (finite) $M \subset \mathbb{N}$ the solution can be obtained in $O(n(ks + n))$ time, where n is the number of input points, $s = |M|$ and k is the number of clusters. This extends an analogous algorithm proposed for the problem assuming the Euclidean norm [4]. It further emphasizes the difference w.r.t. SCC problem, which is NP-hard in dimension 1 for every ℓ_p-norm, showing that relaxing the size constraints is a key condition to guarantee a solution computable in polynomial time. We recall that clustering problems in dimension 1 have already been studied in the literature, especially in connections with problems of computational biology [4, 22]. In particular in [4] an algorithm for solving clustering problem in dimension 1 (with size constraints) is applied for determining promoter regions in genomic sequences, which can be defined intuitively as positions in DNA molecules where the occurrence of certain patterns

of nucleotides allows the cell to active or silence the genes (and hence regulating gene expression).

Other results of the present contribution concern the relaxed size constrained clustering problem in dimension $d = 2$. In this case, assuming ℓ_2-norm and fixing $M = \{2, 4\}$, we prove the problem is strongly NP-hard and it is also easy to see that it does not admit FPTAS unless $P = NP$. Moreover, we prove the same results for the case $M = \{2, 3\}$ both with ℓ_2 and ℓ_1 norm. This implies that also the general relaxed size constrained problem, where M is part of the instance, is strongly NP-hard on the plane both assuming ℓ_2 and ℓ_1-norm.

The introduction of relaxed size constraints is motivated by all applications where one wants to bound the cluster size to certain values, possibly avoiding too large or too small clusters, up to the balanced case where all clusters have almost the same size. Situations of this type are rather common in several contexts [2, 16, 29].

2. Problem definition

In this section we define the problem and fix our notation. Given two positive integers d and p, for 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 $W_p(A_1, ..., A_k)$. [2]

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

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 assuming ℓ_1-norm. This procedure is based on a dynamic programming technique, in the style of [21], and on the so-called String Property [27, 3] (whose definition is recalled below in the proof of Proposition 1). In the case of ℓ_2-norm an analogous algorithm is 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 $(x_i, x_{i+1}, ..., x_j)$. We define the $n \times n$ matrix $U = [U(i, j)]_{i,j=1,\ldots,n}$ by setting $U(i, j) = W_1(X[i, j]) = \sum_{t=i}^{j} |x_t - C_{X[i,j]}|$ 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. *For every instance $(X, k, \mathcal{M})$ of RSC-1 with $|X| = n$, matrix U can be computed in $O(n^2)$ time.*

Proof. For any $1 \leq i \leq j \leq n$ 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 set the left and right sums $L(i, j) := \sum_{i \leq h < m} x_h$ and $R(i, j) := \sum_{m < h \leq j} x_h$. It can be shown that $W_1(X[i, j]) = R(i, j) - L(i, j)$ [3, Prop. 10]. Since $X[i, j]$ is sorted, it can be seen that, for $i < j$,

$$L(i, j) = L(i, j-1) \text{ if } m \in \mathbb{N}, \quad L(i, j) = L(i, j-1) + x_{\lfloor m \rfloor} \text{ otherwise,} \tag{2}$$

$$R(i, j) = R(i, j-1) - x_m + x_j \text{ if } m \in \mathbb{N}, \quad R(i, j) = R(i, j-1) + x_j \text{ otherwise} \tag{3}$$

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 $i \leq j$, can be computed in $O(n^2)$ time, and hence the same holds for determining matrix U. □

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 $h \leq j$, while $Z(h, j) = \infty$ if $h > j$. These values can be derived from U.

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

Proof. Case *i)* is obvious. For $h > 1$ the optimal solution $\{A_1, ..., A_h\}$ for $(X[1, j], h, \mathcal{M})$, satisfies the String Property, i.e. each cluster A_i consists of

consecutive points of X [27, 3]. Then, its right-most cluster A_h has size $|A_h| = m \in \mathcal{M}$ and weight $W_1(A_h) = W_1(X[j-m+1, j]) = U(j-m+1, j)$.

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 $W_1(A_1, ..., A_{h-1}) = \sum_1^{h-1} W_1(A_i) = Z(h-1, j-m)$, 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 1. *RSC-1 with ℓ_1-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 1, matrix $Z = [Z(h, j)]_{h=1,...,k; j=1,...,n}$ 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 considering backward the table of values $\ell_{h,j}$'s for $h \in \{2, \ldots, k\}$, $j \in \{h, \ldots, n\}$.

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 $k < n$. □

Note that the case $s = n$ corresponds to an unconstrained version of our problem. Thus the previous algorithm solves the traditional clustering problem in dimension 1 assuming ℓ_1-norm in time $O(n^2k)$. The same bound holds for ℓ_2-norm [21]. Hence, admitting a small number of possible cardinalities for the clusters reduces the time complexity of the algorithm. However, these polynomial time results only hold for the RSC-1 problem. Indeed, the SCC 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 $\{2, 4\}$-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 $\{2, 4\}$-RSC-2 Problem
Given a point set $X = \{p_1, ..., p_n\} \subset \mathbb{Q}^2$, where n is even, an integer k, $n/4 \leq k \leq n/2$, and a rational value $\lambda > 0$ (threshold), decide whether there exists a $\{2, 4\}$-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 [13]. 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 [12]. Moreover, the edges between variables and clauses can be drawn in a rectilinear fashion [19].

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 $\{2, 4\}$-RSC-2. A similar reduction from Planar 3-SAT to an unconstrained version of the k-means problem in the plane is presented in [18], where however clusters include triples rather than quadruples. Notice that our reduction does not yield multiple copies of the same point in the plane. Another 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.

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 $\{2, 4\}$-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 4, 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 of the embedding of G_Φ (described in details below) is to represent each clause by a square of the grid and associate each variable with a cycle of segments lying on the grid. The cycle of each variable v will enter or go close to every square representing a clause containing v. Since the graph is planar, these cycles can be drawn without crossing; moreover, exactly 3 cycles will enter or go near to each clause-square. The set X includes a pair of points in each clause-square and, for each variable, a family of an even number of points placed along the corresponding cycles. Their position entails the existence of only 2 optimal $\{2\}$-clusterings for the points on each cycle, which may be associated to the possible values $\{0, 1\}$ of the corresponding variable. Thus, it turns out that any clause is satisfiable by an assignment to a variable v if and only if the clause-pair of v can be clustered together with the nearest pair in the optimal $\{2\}$-clusterings associated with the assignment.

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 of these squares.

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

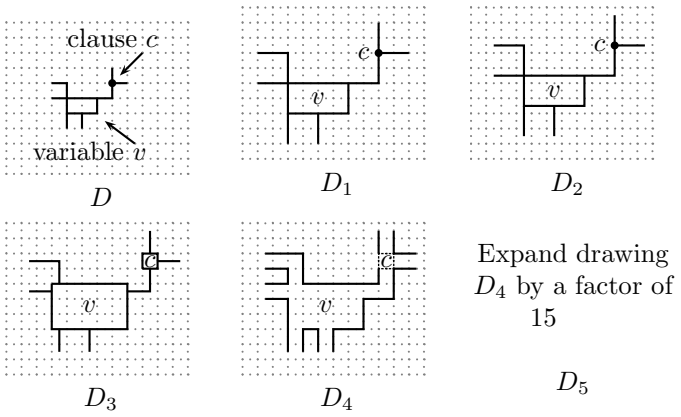

Figure 1: Main steps of the graph transformations used in the reduction.

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 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, we explain later, all $x_{i\ell}$'s lie inside the cycle of drawing D_5 associated with v_i, set at distance 2 from the border, so that each consecutive pair $(x_{it}, x_{i(t+1)})$ forms a segment of length 5.

The precise position of points $x_{i\ell}$'s is illustrated in Figs. 2 and 3, where the cycles are represented by dashed lines and the circuits by continuous lines. It depends on the angles formed between two incident borders. Note that, inside the cycle, every angle has measure either $\pi/2$ (as angle β in Fig. 2) or $3\pi/2$ (as angle α in the same figure); between every pair of angles of measure $\pi/2$ (resp., $3\pi/2$) the position of three consecutive points of the circuit forms two segments of length 5.5 (resp. 4.5). This condition allows to keep the points of the circuits at distance 2 from the border. See for instance, in Fig. 2, points between angles δ and ε, or between angles ι and θ.

Moreover, for every $c_j \in C$, X contains a pair of points u_j, $z_j \in \mathbb{Q}^2$, with the same ordinates, at distance 1 from each other, located near the centre of the square associated with c_j. The exact position of each u_j and z_j is defined by Fig. 3 in the case when two circuits approach the square from right and from left (Γ_i and $\Gamma_{i'}$, respectively), while the third one ($\Gamma_{i''}$) does the same from above (the other cases are symmetric). In this case u_j and z_j are at distance 7 respectively from the left and the right side of the square and both at distance $5 + \delta$ from the upper side, where $\delta > 0$ is a rational value to be fixed later.

In the same picture one can see the position of the points of the circuits near the clause-squares. If Γ_i and $\Gamma_{i'}$ are the circuits approaching the square from right and left, respectively, then they include a vertical segment of length 5 inside the clause-square, at distance 1 from the side of the square; here they

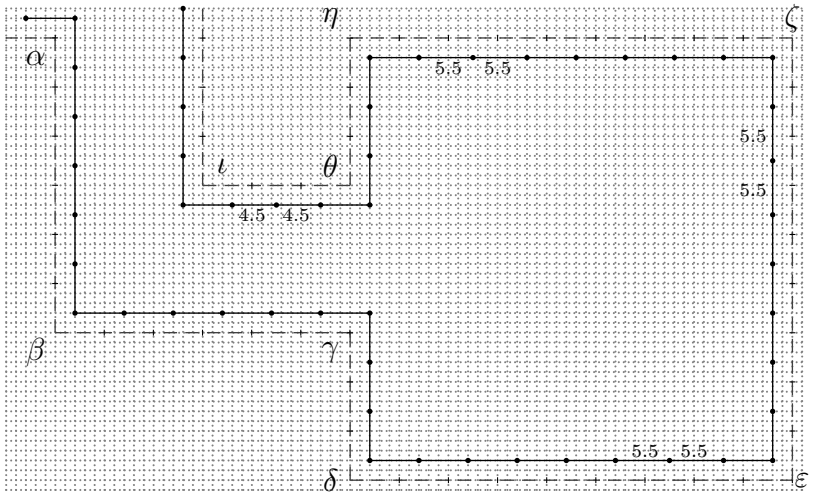

Figure 2: Points of a circuit Γ_i inside the corresponding rectangle. Segments with length different from 5 are indicated. Note that the measure of angles $\alpha, \gamma, \theta, \iota$ is $3\pi/2$, while $\beta, \delta, \varepsilon, \zeta, \eta$ are of measure $\pi/2$.

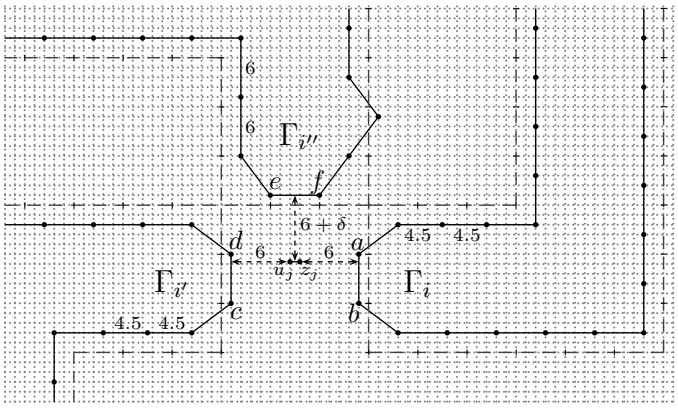

Figure 3: Points of 3 circuits in the neighbourhood of a clause-pair (u_j, z_j). Only length of segments different from 5 are indicated.

are represented by the pairs $\{a, b\}$ and $\{c, d\}$, respectively. One the contrary, on the upper circuit $\Gamma_{i''}$, the segment nearest to the pair u_j, z_j, here represented by $\{e, f\}$, is horizontal and is located outside the clause-square, again at distance 1 from the border. Note that here, all pairs of consecutive points are at distance 5 from each other except before angles of size $3\pi/2$, where triple of points are set to form two consecutive segments of length 4.5 in the circuits Γ_i and $\Gamma_{i'}$, and of length 6 in the upper circuit $\Gamma_{i''}$, respectively.

Thus, we can define the set X by

$$X = \{u_j, z_j \mid j = 1, 2, \ldots, m\} \cup \{x_{i\ell} \mid i = 1, 2, \ldots, n,\ \ell = 1, 2, \ldots, 2L_i\} \quad (4)$$

Now, let us consider the weight of pairs and quadruples in X. First we note that the position of points in any clause-square allows us to evaluate the weight of quadruples Q_1, Q_2, Q_3 defined by

$$Q_1 = \{u_j, z_j, a, b\},\ Q_2 = \{u_j, z_j, c, d\},\ Q_3 = \{u_j, z_j, e, f\}$$

Actually the value of δ can be fixed so that all these clusters have the same weight. In fact, for symmetric reason it is clear that $W_2(Q_1) = W_2(Q_2)$ and, from Equation 1, one can easily verify $W_2(Q_1) = 61.5 - 5\delta + \delta^2$. Analogously, we have $W_2(Q_3) = 49 + 12\delta + \delta^2$ and hence $W_2(Q_1) = W_2(Q_3)$ for $\delta = 25/34$.

In the sequel we assume $\delta = 25/34$ and we call clusters Q_1, Q_2 and Q_3 the *main* quadruples of clause c_j. Moreover, clusters $\{a, b\}$, $\{c, d\}$, and $\{e, f\}$, are called segments *touched* by pair $\{u_j, z_j\}$ in circuits Γ_i, $\Gamma_{i'}$ and $\Gamma_{i''}$, respectively.

Lemma 2. *Under the previous assumptions, the main quadruples of all clauses have the same weight, which is a rational value w such that*

$$58.36 < w < 58.37$$

Moreover, any other cluster of 4 points in X has weight strictly greater than w.

Proof. By the previous discussion it is clear that, assuming $\delta = 25/34$, all the main quadruples of clauses have the same weight w. This value is easily computed from the evaluation of $W_2(Q_1)$ given above, getting $w = \frac{67469}{1156} = 58.36....$

It is also simple to show that the main quadruples of clauses are the unique clusters of 4 elements in X having minimum weight. First note that inside any circuit Γ_i, the quadruple of minimum weight consists of three consecutive segments, one of length 4.5 and two of length 5.5 perpendicular to each other (see Figure 3). Its weight is 81.43.. . Furthermore, another possible quadruple in X can be formed by using the pair u_j, z_j of any clause c_j together with two points belonging to distinct main quadruples at minimal distance from each other. For instance, a cluster of this type is $Q_4 = \{u_j, z_j, f, a\}$ (see again Figure 3). However, also in this case a direct computation shows that $W_2(Q_4) = (223+24\delta+2\delta^2)/4$, which yields $W_2(Q_4)-w > 1/4$. Clearly the other quadruples have larger weight, and hence the main quadruples are those of minimum weight in X. □

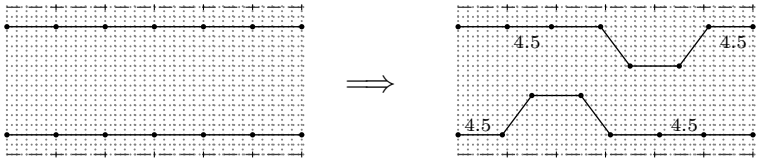

Figure 4: (Left) 30×15 horizontal strip preserving parity. (Right) 30×15 horizontal strip for changing parity. The vertical case is analogous.

Now, let us consider the weight of the pairs of points in X. Note that all pairs of consecutive points in any Γ_i form a segment of possible lengths: 4.5, 5, 5.5, 6, yielding a weight 10.125, 12.5, 15.125, 18, respectively.

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

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

$$\pi_2(i) = \{\{x_{iu}, x_{i(u+1)}\} \mid u = 2, 4, 6, \ldots, 2L_i - 2\} \cup \{\{x_{i(2L_i)}, x_{i1}\}\} \tag{6}$$

3) Parity condition

By a suitable choice of the first point x_{i1}, and possibly by adding new points to Γ_i (as explained below), we can assume that the following parity condition holds: in any Γ_i, for every clause c_j including v_i, the segment touched by (u_j, 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 Γ_i near the clause-square of c_j may be necessary, which are illustrated in Figure 4. This change adds 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).

In passing, we observe that a similar transformation can be used to guarantee that each circuit Γ_i has an even number of points $(2L_i)$.

4) Definition of k and λ

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

$$\lambda = wm + \frac{5^2}{2}(k - h) + \frac{1}{2}\left[\frac{4.5^2}{2} \cdot s_{4.5} + \frac{5.5^2}{2} \cdot s_{5.5} + \frac{6^2}{2} \cdot s_6\right]$$

where w is defined as in Lemma 2, s_u is the total number of segments of length u in all Γ_i's for $u \in \{4.5, 5.5, 6\}$, and $h = m + \frac{1}{2}(s_{4.5} + s_{5.5} + s_6)$.

One can show that every $\{2, 4\}$-clustering π of X into k clusters must contain exactly m quadruples. Indeed, if n_Q and n_S denote, respectively, the number of quadruples and the number of pairs of π, then $|X| = 2n_S + 4n_Q$ and $k = n_S + n_Q$, which yields $n_Q = m$ and $n_S = k - m$. By Lemma 2 in π all m quadruples have weight at least w. Moreover, by construction of circuits Γ_i's, π may include

at most $s_u/2$ many segments of length u for each $u \in \{4.5, 5.5, 6\}$ 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 Φ is satisfiable if and only if there exists a $\{2,4\}$-clustering of X of weight at most λ, consisting of k clusters. Suppose Φ 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 pair $\{u_j, z_j\}$ can be clustered together with the touched segment in Γ_i, for a variable v_i whose assignment satisfies 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 quadruples 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 $\{2,4\}$-clustering of X of weight λ having k clusters.

Vice-versa, if there exists a $\{2,4\}$-clustering π of X with k clusters and weight λ, then such a clustering must contain m quadruples of weight w. The only way to obtain these quadruples is to cluster each pair $\{u_j, z_j\}$ with a touched segment of a circuit Γ_i approaching the clause-square of c_j. By the parity condition this defines an assignment of values to all variables that makes true each clause of Φ.

Theorem 2. *Assuming ℓ_2-norm, the $\{2,4\}$-RSC-2 problem is strongly NP-hard and it does not admit an FPTAS unless $P = NP$. This implies the same result for the general 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 [26, Sec. 8.3], the same problem does not admit an FPTAS unless $P = NP$. □

5. NP-hardness of $\{2,3\}$-RSC-2 with ℓ_2 norm

In this section we prove another NP-hardness result concerning the $\{2,3\}$-RSC in the Euclidean plane. The decision problem can be stated as follows: Given a point set $X = \{p_1, ..., p_n\} \subset \mathbb{Q}^2$, an integer k, $n/3 \leq k \leq n/2$, and a rational value $\lambda > 0$, decide whether there exists a $\{2,3\}$-clustering $\{A_1, ..., A_k\}$ of X, such that $W_2(A_1, ..., A_k) \leq \lambda$. Similarly to the previous case all solutions, for a given instance, have the same number pairs and the same number of triples (rather than quadruples), which now are $n_s = 3k - n$ and $n_t = n - 2k$, respectively.

Again the reduction is obtained from Planar 3-SAT and is based on the construction described in Section 4 with some changes. Now, all quadruples are replaced by triples and the main difference is that each clause is represented by a single point in the corresponding square, rather than a pair of points.

More precisely, given a planar 3-CNF formula Φ of m clauses and n variables, the graph G_Φ is embedded into a planar grid as in Section 4 (Steps 0, 1,...,5). Again, the point set X includes, for each variable v_i of Φ, a circuit Γ_i of $2L_i$ points for some integer L_i. Moreover, for every clause c_j of Φ, X includes a point z_j placed near the centre of the corresponding 15×15 clause-square, at distance 7.5 from the left and right side of the square and at distance $5 + 23/30$ from its upper side (see Fig. 5). Thus, we have

$$X = \{z_j \mid j = 1, 2, \ldots, m\} \cup \{x_{i\ell} \mid i = 1, 2, \ldots, n,\ \ell = 1, 2, \ldots, 2L_i\} \quad (7)$$

The points of the circuits Γ_i's are placed as in the previous section except near the clause-square, where they are set as shown in Fig. 5. This picture shows three circuits Γ_i, $\Gamma_{i'}$ and $\Gamma_{i''}$ approaching point z_j, respectively from right, left and from above. Note that the location of points of $\Gamma_{i''}$ is now symmetric and no segment of length 6 appears in the circuits. Moreover, the segment of Γ_i nearest to point z_j is inside the clause-square, at distance 2 from the border (rather than 1); the same occurs for $\Gamma_{i'}$, while the nearest segment of $\Gamma_{i''}$ is just on the side of the square. As in the previous section we say that these segments are *touched* by point z_j. If $\{a, b\}$ is a segment touched by z_j then the cluster $\{z_j, a, b, \}$ is called a *main* triple of z_j.

Thus, a natural $\{2, 3\}$-clustering of X into k sets can be built by taking a main triple for each clause c_j and the other $k - m$ pairs by choosing, for each set Γ_i, the $\{2\}$-clustering $\pi_1(i)$ or $\pi_2(i)$ defined as in (5) and (6).

By direct computation one can prove, assuming ℓ_2 norm, that all main triples have the same weight, given by the value $w' = \frac{23402}{675}$, which satisfies

$$34.66 < w' < 34.67$$

The only triples that have a weight smaller than w' are the sets of three consecutive points, in any circuit Γ_i, that form a right triangle, i.e. three points $\{x_{i(r-1)},\ x_{ir}, x_{i(r+1)}\}$ such that the segments $(x_{i(r-1)}, x_{ir})$ and $(x_{ir}, x_{i(r+1)})$ are of length 5 and are perpendicular. In this case the weight of the cluster $\{x_{i(r-1)}, x_{ir}, x_{i(r+1)}\}$ is $100/3$, which is smaller than w'. However, replacing a main triple of a point z_j with such a right triangle forces to replace a segment of length 5 with a pair of points of weight $[5.5^2 + (23/30)^2]/2 = 15.41...$ (that one formed by z_j with a nearest point of some circuit). Thus, the overall weight of the clustering increases since $12.5 + w' < 15.4 + 100/3$.

The parity condition is here defined as in the previous section, as well as the value of k, again set to $k = \sum_1^n L_i$, while λ is given by

$$\lambda = \frac{5^2}{2}(k - h) + w'm + \frac{1}{2}\left[\frac{4.5^2}{2} \cdot s_{4.5} + \frac{5.5^2}{2} \cdot s_{5.5}\right]$$

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

The remaining part of the discussion of the previous section holds almost unchanged. Thus, it turns out that Φ is satisfiable if and only if there exists a $\{2, 3\}$-clustering of X consisting of k sets of total weight λ.

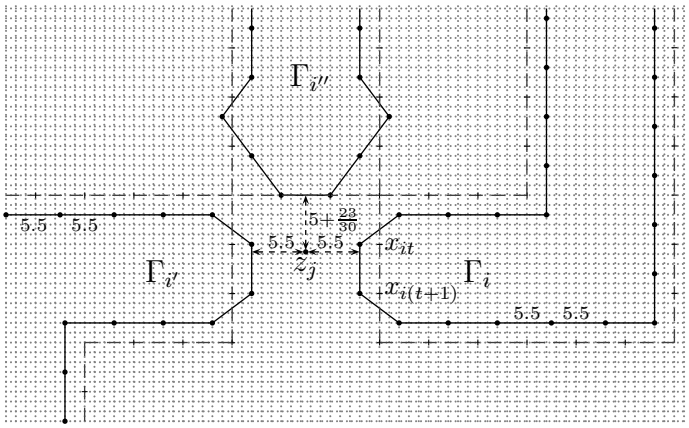

Figure 5: Points of 3 circuits in the neighbourhood of a clause-point z_j. Only length of segments different from 5 are indicated.

Theorem 3. *Assuming ℓ_2-norm, the $\{2,3\}$-RSC-2 problem is strongly NP-hard and it does not admit an FPTAS unless $P = NP$.*

6. NP-hardness of RSC-2 with ℓ_1-norm

The same result for $\{2,3\}$-RSC-2 can be obtained assuming ℓ_1-norm. In this case the reduction from Planar 3-SAT can be simplified. The differences w.r.t. the construction presented in Sections 4 and 5 are the following:

1. In the definition of the planar grid, at step 5, the drawing is expanded by a factor of 6 rather than 15.
2. Every clause-point z_j is now located in the centre of the (6×6) square associated with clause c_j.
3. For every variable v_i the corresponding circuit Γ_i consists of points placed inside the cycle of v_i at distance 1 from the border, while every two consecutive points are at distance 2 from each other (in ℓ_1 norm); their position, in particular near the angles of the cycle, is described in Figure 6.
4. $\lambda = 6m + 2(k - m)$.
5. The parity condition can be guaranteed by simple modifications to any circuit Γ_i, if necessary, similar to the transformation described in Figure 4. The resulting circuit may have two more points, one before and one after any touched segment, so that the ℓ_1-distance between consecutive points remains 2.

Note that now every triple formed by a clause-point z_j and a touched segment has weight 6, while every triple of consecutive points in each circuit has weight 4. However, replacing a triple of the first type with one of the second, in a given $\{2,3\}$-clustering, forces to add a new segment of weight 5 (joining z_j with an endpoint of the touched segment) reducing by 1 the number of pairs of weight 2. Thus, such a replacement increases the overall weight of the clustering. This

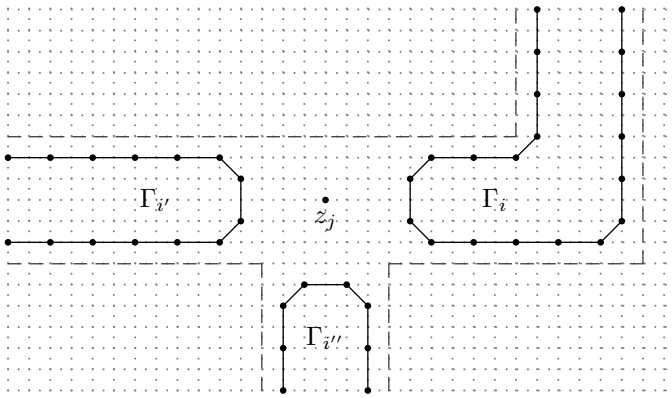

Figure 6: Points of 3 circuits in the neighbourhood of a clause-point. All segments formed by consecutive points have ℓ_1-length 2.

shows that a solution including m triples of weight 6, each consisting of a clause-point with one of its touched segments, and $k - m$ pairs of consecutive points of circuits Γ_i's, yields a $\{2, 3\}$-clustering of minimum weight.

The other details and reasoning of Section 4 remains unchanged and this proves the following

Theorem 4. *Assuming ℓ_1-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.*

7. 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-norm. As a similar procedure solves the problem assuming ℓ_2-norm, a natural question is whether an analogous polynomial time algorithm exists for every ℓ_p-norm with integer $p > 2$. We recall that the clustering in dimension 1 is motivated by bioinformatics applications as illustrated in [4, 22].

We observe that there exist polynomial time algorithms for $\mathcal{M}$-RSC-d problems even in a dimension $d > 1$. For instance, the $\{2\}$-RSC-d 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 assuming any ℓ_p-norm, by using classical algorithms [20]. The same occurs fixing $\mathcal{M} = \{1, 2\}$ 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 [23, sec. 3.1.1]). Another problem solvable in polynomial time, both with ℓ_1 and ℓ_2 norm, is to find the best bipartition of n points in the plane into two clusters of size m and $n - m$ respectively, where m is part of the instance [14]. This includes both the balanced 2-clustering problem in the plane [2] and our RSC-2

problem restricted to the case $k = 2$. More generally, we conjecture that RSC-2 problem restricted to fixed $k > 2$ is also solvable in polynomial time. For these reasons, while the unconstrained clustering problem in dimension 2 is NP-hard [18], the relaxed constrained versions are not always difficult.

On the other hand, as shown in our work, such a difficulty occurs in some situations where the number of clusters remains unbounded. We have proved that, assuming ℓ_2-norm, $\{2, 4\}$-RSC-2 problem is strongly NP-hard and the same occurs for $\{2, 3\}$-RSC-2 both with ℓ_1 and ℓ_2-norm. These negative results lead to conjecture that, in dimension 2, the problem remains NP-hard whenever the set $\mathcal{M}$ of size constraints includes a fixed value greater than 2. Thus, even if the situation is not uniform, we believe that in most cases our relaxed constrained clustering problem remains difficult in dimension 2.

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, Dept. 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] M. Goldwurm, J. Lin, and F. Saccà. On the complexity of clustering with relaxed size constraints. In *Algorithmic Aspects of Information and Management (Proc. 11th AAIM)*, volume 9778 of *LNCS*, pages 26–38. Springer, 2016.

[11] 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.

[12] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[13] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[14] J. Lin, A. Bertoni, and M. Goldwurm. Exact algorithms for size constrained 2-clustering in the plane. *Theoretical Computer Science*, 629:80–95, 2016.

[15] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[16] G. Ma, J. Peng, and Y. Wei. On approximate balanced bi-clustering. In *Proc. International Computing and Combinatorics Conference (COCOON 2005)*, volume 3595 of *LNCS*, pages 661–670. Springer, 2005.

[17] J. B. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. on Math. Struct.*, volume 1, pages 281–297, 1967.

[18] M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.

[19] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2), 2008.

[20] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[21] M. R. Rao. Cluster Analysis and Mathematical Programming. *Journal of the American Statistical Association*, 66(335):622–626, 1971.

[22] C. Schmid, T. Sengstag, P. Bucher, and M. Delorenzi. Madap, a flexible clustering tool for the interpretation of one-dimensional genome annotation data. *Nucleic Acids Research*, 35:W201–W205, 2007.

[23] R. Stephan. Cardinality constrained combinatorial optimization: Complexity and polyhedra. *Discrete Optimization*, 7(3):99–113, 2010.

[24] 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.

[25] A. Vattani. k-means Requires Exponentially Many Iterations Even in the Plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011.

[26] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[27] H. Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64(326):506–519, 1969.

[28] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. 17th Intl. Conf. on Machine Learning*, pages 1103–1110, 2000.

[29] S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.