UNIVERSITÀ DEGLI STUDI DI MILANO

SCUOLA DI DOTTORATO IN SCIENZE MATEMATICHE

DIPARTIMENTO DI MATEMATICA "FEDERIGO ENRIQUES"

CORSO DI DOTTORATO DI RICERCA IN
MATEMATICA E STATISTICA PER LE SCIENZE COMPUTAZIONALI
XXIII CICLO

EXACT ALGORITHMS FOR SIZE CONSTRAINED CLUSTERING

Settori scientifico-disciplinari: INF/01, MAT/03

Author
Jianyi LIN

Supervisor
Prof. Alberto BERTONI

Ph.D. Program Coordinator
Prof. Giovanni NALDI

Academic Year 2010–11

Abstract This thesis investigates the following general constrained clustering problem: given a dimension d, an L^p-norm, a set $X \subset \mathbb{R}^d$, a positive integer k and a finite set $\mathcal{M} \subset \mathbb{N}$, find the optimal k-partition $\{A_1, ..., A_k\}$ of X w.r.t. the L^p-norm satisfying $|A_i| \in \mathcal{M}$, $i = 1, ..., k$.
First of all, we prove that the problem is NP-hard even if $k = 2$ (for all $p > 1$), or $d = 2$ and $|\mathcal{M}| = 2$ (with Euclidean norm). Moreover, we put in evidence that the problem is computationally hard if p is a non-integer rational.
When $d = 2$, $k = 2$ and $\mathcal{M} = \{m, n - m\}$, we design an algorithm for solving the problem in time $O(n \sqrt[3]{m} \log^2 n)$ in the case of Euclidean norm; this result relies on combinatorial geometry techniques concerning k-sets and dynamic convex hulls.
Finally, we study the problem in fixed dimension d with $k = 2$; by means of tools of real algebraic geometry and numerical techniques for localising algebraic roots we construct a polynomial-time method for solving the constrained clustering problem with integer p given in unary notation.

Contents

Introduction

Clustering or cluster analysis [5] is a method in unsupervised learning and one of the most used techniques in statistical data analysis. Clustering has a wide range of applications in many areas like pattern recognition, medical diagnostics, data mining, biology, market research and image analysis among others. A cluster is a set of data points that in some sense are similar to each other, and clustering is a process of partitioning a data set into disjoint clusters. In *distance clustering*, the similarity among data points is obtained by means of a *distance* function.

Fixed a norm $\| \|_p$ $(p \geq 1)$, given a point set $X \subset \mathbb{Q}^d$ and an integer k, clustering problem consists in finding a k-partition $\{A_1, ..., A_k\}$ of X that minimises the function

$$W(A_1, ..., A_k) = \sum_{i=1}^{k} \sum_{x \in A_i} \|x - C_i\|_p^p$$

where C_i is the p-centroid of A_i, i.e.

$$C_i = \arg\min_{\mu} \sum_{x \in A_i} \|x - \mu\|_p^p$$

Distance clustering is a difficult problem. For an arbitrary dimension d the problem is NP-hard even if the number k of clusters equals 2 [1]; the same occurs if $d = 2$ and k is arbitrary [8, 12]. For the Euclidean distance, a well-known heuristic is Lloyd's algorithm [6, 7], also known as the k-Means Algorithm; since this is a heuristic procedure, there is no guarantee that it converges to the global optimum. This algorithm is usually very fast, but it can require exponential time in the worst case [11].

In real-world problems, often people have some information on the clusters: incorporating this information into traditional clustering algorithms can increase the clustering performance. Problems that include background information are called *constrained clustering* problems and are divided in two classes.

On the one hand, clustering problems with instance-based constraints typically comprise a set of must-link constraints or cannot-link constraints [13], defining pairs of elements that must be included, respectively, in the same cluster or in different clusters.

On the other hand, clustering problems with cluster-based constraints [2, 10] incorporate constraints concerning the size of the possible clusters. Recently, in [14] cluster size constraints are used for improving clustering accuracy; this approach, for instance, allows one to avoid extremely small or large clusters in standard cluster analysis.

In this work we consider two types of problems:
• Size Constrained Clustering Problem (SCC):
Given a point set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{Q}^d$, 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

$$|A_i| = m_i \quad \text{for } i = 1, ..., k$$

that minimizes the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

• Relaxed Constraints Clustering Problem (RCC):
Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^d$, an integer $k > 1$ and a set $\mathcal{M} = \{m_1, m_2, ..., m_s\}$ of positive integers, find a k-clustering $\{A_1, ..., A_k\}$ with

$$|A_i| \in \mathcal{M} \quad \text{for all } i = 1, ..., k$$

that minimises the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

The main result presented in Chapter 1 (Theorem 1.4) is a property verified by the optimal solution of an instance of SCC with $k = 2$: if $\{A, \bar{A}\}$ is the optimal 2-partition of X, then A is separated from $\bar{A}$ by an hypersurface of the kind:

$$\|x - \alpha\|_p^p - \|x - \beta\|_p^p = c$$

In this work we show that SCC is a difficult problem. The main hardness results we obtain in Chapter 2 are:

1) For every norm $\| \ \|_p$ with $p > 1$, RCC with clustering size k fixed is NP-hard, even in the case $k = 2$ and $\mathcal{M} = \{\frac{n}{2}\}$ (Theorem 2.2).

2) For every norm $\| \ \|_p$ with $p \geq 1$, SCC with dimension d fixed is NP-hard, even in the case $d = 1$ (Theorem 2.5). Observe that RCC in dimension 1 is solvable in polynomial time [9].
3) For the euclidean norm $\| \ \|_2$, RCC in dimension $d = 2$ is NP-complete even if the possible size constraints are $\{2, 3\}$ (Theorem 2.10)

For illustrating some subtleties in the case of $\| \ \|_p$ with non-integer rational p, we consider the problem p-LC of localising the centroid of integers set $\{x_1, ..., x_n\}$ (i.e. $d = 1$).
We prove that:

4) SQRT-Sum is polynomially reducible to $\frac{3}{2}$-LC (Theorem 2.4).

This puts in evidence that it is questionable whether p-LC is in NP.

Since we prove that the RCC problem in the plane with constraints $\{2, 3\}$ is NP-complete, we can't expect to obtain an exact algorithm for the general RCC problem in the plane.

In Chapter 3 we investigate RCC in the plane with a fixed clustering size $k = 2$. In particular, we consider the problems:
• 2-RCC in the Plane (briefly 2-RCC):
Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$ and $\mathcal{M} = \{k, n-k\}$, find a 2-clustering $\{A, \bar{A}\}$ of X with $|A| = k, |\bar{A}| = n - k$, that minimises

$$W(A, \bar{A}) = W(A) + W(\bar{A})$$

• Full 2-RCC in the plane (briefly Full 2-RCC):
Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$, find all the optimal 2-clusterings $\pi_k = \{A_k, \bar{A}_k\}$, with $|A_k| = k$ for all k, $1 \leq k \leq \lfloor \frac{|X|}{2} \rfloor$.

The main results we obtain are:

1) There is an algorithm for solving Full 2-RCC problem in time $O(n^2 \cdot \log n)$ (Theorem 3.2).
2) There is an algorithm for solving 2-RCC problem in time $O(n\sqrt[3]{k} \cdot \log^2 n)$ (Theorem 3.17).

It should be observed that, the algorithm to solve 2-RCC requires methods related to the challenging problem [4] of Combinatorial Geometry of enumerating the k-sets of points X in the plane.

At the end, in Chapter 4 we study the problem 2-SCC in fixed dimension d. We will prove that, by appropriately decomposing the space of the parameter of the hypersurfaces separating the 2 clusters, we obtain a set of 2-clusterings containing the optimal clustering of size m, for every constraint size m, thus allowing us to compute the optimal costs and the cardinality of the clusters, for every cluster size constraint m. This method will be proved to have polynomial time complexity with respect to the data set size n and the integer p, for a fixed dimension d.

It is noteworthy to anticipate that in Chapter 4 we widely make use of concepts and methods from Real Algebraic Geometry; in particular we apply the cylindrical algebraic decomposition [3] for solving the 2-SCC problem.

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. P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained K-Means Clustering. Technical Report MSR-TR-2000-65, Miscrosoft Research Publication, May 2000.
3. G. E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In E. Barkhage, editor, *Proc. 2nd GI Conf. on Automata Theory and Formal Lang.*, volume 33 of *LNCS*, pages 134–183, Berlin, 1975. Springer.
4. P. Erdős, L. Lovász, A. Simmons, and E. G. Straus. Dissection graphs of planar point sets. In *A survey of combinatorial theory (Proc. Internat. Sympos., Colorado State Univ., Fort Collins, Colo., 1971)*, pages 139–149. North-Holland, Amsterdam, 1973.
5. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag, 2nd edition, 2009.
6. S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
7. J. B. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Structures*, pages 281–297, 1967.
8. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The Planar k-Means Problem is NP-Hard. In S. Das and R. Uehara, editors, *WALCOM: Algorithms and Computation*, volume 5431 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin/Heidelberg, 2009.
9. F. Saccà. *Problemi di Clustering con Vincoli: Algorithmi e Complessità.* PhD thesis, University of Milan, Milan, 2010.
10. 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 *Lecture Notes in Computer Science*, pages 405–419. Springer Berlin/Heidelberg, 2001.
11. A. Vattani. K-means requires exponentially many iterations even in the plane. In *Proceedings of the 25th Symposium on Computational Geometry (SoCG)*, 2009.
12. A. Vattani. The hardness of k-means clustering in the plane. manuscript, 2009.
13. K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. of the 17th Intl. Conf. on Machine Learning*, pages 1103–1110, 2000.
14. S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.

Chapter 1
Clustering problems and preliminary results

1.1 Introduction

Clustering is one of most used technique in statistical data analysis, with applications in areas like pattern recognition, data mining, image analysis among others [2]. A cluster is a set of data points similar, and clustering is a process of partitioning a data set into disjoint clusters. In distance clustering the similarity among points is given by means of a distance function.

Fixed a norm $\| \|_p$ $(p \geq 1)$, given a point set $X \subset \mathbb{Q}^d$ and an integer k, clustering problem consists in finding a k-partition $\{A_1, ..., A_k\}$ of X that minimises the function

$$W(A_1, ..., A_k) = \sum_{i=1}^{k} \sum_{x \in A_i} \|x - C_i\|_p^p$$

where C_i is the p-centroid of A_i, i.e.

$$C_i = \arg\min_{\mu} \sum_{x \in A_i} \|x - \mu\|_p^p$$

In the real world problems, often people have some information on the clusters: problems that include such information are called constrained clustering. In this work we consider constraints concerning the size of the possible clusters [3, 6, 8].

We will consider 2 kinds of problems, formally stated in Section 1.2.
In the first one, called Size Constrained Clustering Problem (SCC), it is given in input $X \subset \mathbb{Q}^d$, an integer k and a vector $(j_1, ..., j_k)$ of positive integers s.t. $\sum j_i = |X|$. In this case, an admissible solution is a k-partition $\{A_1, ..., A_k\}$ of X s.t.

$$|A_1| = j_1, ..., |A_k| = j_k.$$

In the second kind of problem, called Relaxed Constraints Clustering (RCC), it is given in input X, k and a set $\mathcal{M} = \{g_1, ..., g_s\}$ of integers. An admissible solution is a k-partition $\{A_1, ..., A_k\}$ of X s.t.

$$|A_j| \in \mathcal{M} \qquad \text{for all } 1 \leq j \leq k.$$

The main result of this chapter (Theorem 1.4) is a property (Separation Property), verified by the optimal solution of an instance of SCC with $k = 2$: if $\{A, \bar{A}\}$ is the optimal 2-partition of X, then A is separated from $\bar{A}$ by an hypersurface of the kind:

$$\|x - \alpha\|_p^p - \|x - \beta\|_p^p = c$$

In Section 1.4 some consequences of the Separation property in 1-dimensional case (i.e. $X \subset \mathbb{R}^1$) are studied. In Theorem 1.6 we extend to the constrained clustering a property observed in the clustering problem by Fisher [4] in case of Euclidean norm, and by Novick [5] in case of norm $\| \ \|_p$, with $p > 1$.

At the end, in Section 1.5, it is observed that the p-centroid (integer $p > 1$) C of a set $X \subset \mathbb{Q}^1$ is not in general a rational number, and a method for approximating C by a rational number is given.

Part of the results of this chapter are published in [1].

1.2 Constrained Clustering Problems

In this section we introduce formally the problem of Size Constrained Clustering, that will be studied in this work from the point of view of computational complexity of exact algorithms for solving it.

Hereafter, for a positive integer d, we consider the space $\mathbb{R}^d$ equipped with the p-norm denoted by $\| \cdot \|_p$, with fixed $p \geq 1$, where $\|(\alpha_1, \alpha_2, ..., \alpha_d)\|_p = (\sum |\alpha_i|^p)^{\frac{1}{p}}$.

Let $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{Q}^d$. A *k-clustering* is a k-partition of X, i.e. a family $\{A_1, A_2, ..., A_k\}$ of k nonempty subsets of X such that $\bigcup_{i=1}^k A_i = X$ and $A_i \cap A_j = \emptyset$, for $i \neq j$. Every A_i is called a *cluster*. The *p-centroid* (or simply *centroid* when p is clearly understood) C_A of a cluster $A \subseteq X$ is

$$C_A = \arg\min_{\mu \in \mathbb{R}^d} \sum_{x \in A} \|x - \mu\|_p^p$$

If $p > 1$ it is well-known that the centroid is unique in one-dimensional case [5], and remains unique in multi-dimensional case by easily treating each component separately; in particular when $p = 2$ the centroid is the mean

$C_A = (\sum_{x\in A} x)/|A|$. In the case $p = 1$ we can have different centroids; one of them is the componentwise median.

The *cost* $W(A)$ of a cluster A is

$$W(A) = \sum_{x\in A} \|x - C_A\|_p^p \tag{1.1}$$

while the *cost* of a k-clustering $\{A_1, A_2, ..., A_k\}$ is $W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i)$. The classical *Clustering Problem* is formulated as follows.

Definition 1.1 (Clustering Problem). Given a point set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{Q}^d$ and an integer $k > 1$, find a k-clustering $\{A_1, A_2, ..., A_k\}$ that minimizes the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

In this work we are interested in a version of clustering problem, where the cardinalities of the clusters are constrained. Formally, the problem can be stated as follows:

Definition 1.2 (Size Constrained Clustering Problem (SCC)). Given a point set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{Q}^d$, 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

$$|A_i| = m_i \quad \text{for } i = 1, ..., k$$

that minimizes the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

Moreover, we relaxed the size constraints of the clustering, thus defining a weaker version of SCC, as follows.

Definition 1.3 (Relaxed Constraints Clustering Problem (RCC)). Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^d$, an integer $k > 1$ and a set $\mathcal{M} = \{m_1, ..., m_s\}$ of positive integers, finda a k-clustering $\{A_1, ..., A_k\}$ with

$$|A_i| \in \mathcal{M} \qquad \text{for all } i = 1, ..., k$$

that minimises the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

W.l.o.g. we can make the assumption that $X = \{x_1, ..., x_n\}$ is composed by vectors of positive integer coordinates represented in binary notation, whose size is $\sum |x_k|_b$, where $|x_k|_b$ is the number of bits of $x_k \in \mathbb{N}^d$.

Observe that this is equivalent to considering $\{x_1, ..., x_n\} \subset \mathbb{Q}^d$, the set of rational coordinate points, since the solution of the problems is invariant to translating and scaling. In fact, the instances

$$X_1 = \{x_1, ..., x_n\}$$
$$X_2 = \{x_1 + a, ..., x_n + a\}$$
$$X_3 = \{\lambda x_1, ..., \lambda x_n\}$$

do admit the same optimal solution. When $d = 1$ we can also assume that X is composed by positive integers $x_1 < x_2 < ... < x_n$.

We stress that in the SCC problem the integers n, k and d are part of the instance. On the contrary, if d is fixed the problem is called SCC-d; if k is fixed the problem is called k-SCC; furthermore, if both d and k are fixed the problem is called k-SCC-d. However, when it is clear from the context which are the fixed parameters, we will simply write SCC (RCC) instead of k-SCC, k-RCC and so on.

1.3 Separation results

In this section we prove a separation property that is verified for the optimal solution of 2-SCC. We first need a simple lemma stating that if $p > 1$ then the centroid of a set of points moves whenever one of the points changes. The property is not true in the case $p = 1$.

Lemma 1.1. Given $n+1$ reals $x_1, x_2, ..., x_n, \bar{x}_1$ and $p > 1$, let $C(x_1, x_2, ..., x_n)$ be the centroid of $\{x_1, x_2, ..., x_n\}$ and $C(\bar{x}_1, x_2, ..., x_n)$ be the centroid of $\{\bar{x}_1, x_2, ..., x_n\}$. If $\bar{x}_1 \neq x_1$, then $C(x_1, x_2, ..., x_n) \neq C(\bar{x}_1, x_2, x_3, ..., x_n)$.

Proof. Let's suppose that $C(x_1, x_2, ..., x_n) = C(\bar{x}_1, x_2, x_3, ..., x_n) = C$. Setting $F(\mu) = \sum_i |x_i - \mu|^p$, since $F(\mu)$ is strictly convex [5], it follows that $F'(C) = 0 = \sum \mathrm{sgn}(x_i - C)|x_i - C|^{p-1}$. Analogously, we have $0 = \mathrm{sgn}(\bar{x}_1 - C)|\bar{x}_1 - C|^{p-1} + \sum_2^n \mathrm{sgn}(x_i - C)|x_i - C|^{p-1}$. This implies that $\mathrm{sgn}(\bar{x}_1 - C)|\bar{x}_1 - C|^{p-1} = \mathrm{sgn}(x_1 - C)|x_1 - C|^{p-1}$, that is $x_1 = \bar{x}_1$. □

Corollary 1.2. Fixed $p > 1$, let C be the centroid of $\{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$ and $\bar{C}$ the centroid of $\{\bar{x}_1, x_2, x_3, ..., x_n\} \subset \mathbb{R}^d$, where $\bar{x}_1 \neq x_1$. Then:

$$\sum_{i=1}^{n} \|x_i - C\|_p^p < \sum_{i=1}^{n} \|x_i - \bar{C}\|_p^p$$

Proof. Since $\bar{x}_1 \neq x_1$ there is a component (say ℓ, with $1 \leq \ell \leq d$) of x_1 different from the corresponding component of $\bar{x}_1$. Notice that the ℓ-component of the centroid of a point set depends only on the ℓ-component of these points. By Lemma 1.1, the ℓ-component of C is different from the ℓ-component of $\bar{C}$, hence $C \neq \bar{C}$. Since C is the unique minimum point of the function $\sum_i \|x_i - \mu\|_p^p$, the thesis follows. □

Proposition 1.3. Fixed $p > 1$, let $\{A, B\}$ be the optimal solution of a 2-SCC problem on the instance $X \subset \mathbb{R}^d$ with $|A| = k$. If $x \in A$ and $y \in B$, it holds:

$$\|x - C_A\|_p^p + \|y - C_B\|_p^p < \|x - C_B\|_p^p + \|y - C_A\|_p^p$$

Proof. Since $\{A, B\}$ is a partition, then $x \neq y$. Suppose by contradiction that:

$$\|x - C_A\|_p^p + \|y - C_B\|_p^p \geq \|x - C_B\|_p^p + \|y - C_A\|_p^p \tag{1.2}$$

For $S \subset X$ we set $F_S(\mu) = \sum_{x \in S} \|x - \mu\|_p^p$. Then, we obtain:

$$\begin{aligned}
W(A, B) &= F_A(C_A) + F_B(C_B) \\
&= F_{A \smallsetminus \{x\}}(C_A) + \|x - C_A\|_p^p + F_{B \smallsetminus y}(C_B) + \|y - C_B\|_p^p \\
&\geq F_{A \smallsetminus \{x\}}(C_A) + \|y - C_A\|_p^p + F_{B \smallsetminus \{y\}}(C_B) + \|x - C_B\|_p^p \text{ (by (1.2))} \\
&= F_{A \smallsetminus \{x\} \cup \{y\}}(C_A) + F_{B \smallsetminus \{y\} \cup \{x\}}(C_B) \\
&> F_{A \smallsetminus \{x\} \cup \{y\}}(C_{A \smallsetminus \{x\} \cup \{y\}}) + F_{B \smallsetminus \{y\} \cup \{x\}}(C_{B \smallsetminus \{y\} \cup \{x\}}) \text{ (by Cor. 1.2)} \\
&= W(A \smallsetminus \{x\} \cup \{y\}, B \smallsetminus \{y\} \cup \{x\})
\end{aligned}$$

This is a contradiction, since $A \neq A \smallsetminus \{x\} \cup \{y\}$, but $|A| = |A \smallsetminus \{x\} \cup \{y\}| = k$. This would imply that $\{A, B\}$ is not the optimal solution. □

Theorem 1.4 (Separation Property). Fixed $p > 1$, let $\{A, B\}$ be an optimal solution of a 2-SCC on the instance $\{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$ with size constraint $|A| = k$. Then we have that:

1. $C_A \neq C_B$
2. there exists $c \in \mathbb{R}$ such that:

$$\begin{aligned}
x \in A &\text{ implies } \|x - C_A\|_p^p - \|x - C_B\|_p^p < c \\
x \in B &\text{ implies } \|x - C_A\|_p^p - \|x - C_B\|_p^p > c
\end{aligned}$$

Proof. We notice that, by Proposition 1.3, if $x_i \in A$ and $x_j \in B$ it holds:

$$\|x_i - C_A\|_p^p - \|x_i - C_B\|_p^p < \|x_j - C_A\|_p^p - \|x_j - C_B\|_p^p \tag{1.3}$$

Since $x_i \neq x_j$ it follows that $C_A \neq C_B$, otherwise (1.3) yields $0 < 0$. Let $\alpha = \max_{x \in A} \|x - C_A\|_p^p - \|x - C_B\|_p^p$ and $\beta = \min_{x \in B} \|x - C_A\|_p^p - \|x - C_B\|_p^p$. By (1.3) we obtain $\alpha < \beta$. Setting $c = \frac{\alpha + \beta}{2}$, it holds $\alpha < c < \beta$, hence:

$$x \in A \text{ implies } \|x - C_A\|_p^p - \|x - C_B\|_p^p \leq \alpha < c$$
$$x \in B \text{ implies } \|x - C_A\|_p^p - \|x - C_B\|_p^p \geq \beta > c$$

□

The previous theorem states that, in $\mathbb{R}^d$ the hypersurface of equation

$$\|x - C_A\|_p^p - \|x - C_B\|_p^p = c \tag{1.4}$$

is well-defined and strictly separates the sets A and B of an optimal solution. In the particular case $p = 2$, the hypersurface becomes a hyperplane; in fact we have that (1.4) reduces to

$$\langle x, (C_B - C_A) \rangle = \frac{c + \|C_B\|_2^2 - \|C_A\|_2^2}{2}$$

which is the equation of the *bisecting plane* in $\mathbb{R}^d$ (here $\langle \cdot, \cdot \rangle$ denotes the scalar product).

1.4 One-dimensional case: String Property

In this section we consider the case $d = 1$, i.e. $X = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}$ for each i, and we show a structural property (usually named String Property, a term coined by Vinod [7] and used in literature) of the optimal size constrained k-clustering. In this way we extend to the constrained clustering a property observed in the clustering problem by Fisher [4] in the case $p = 2$, and Novick [5] in the case $p > 1$.

Definition 1.4. A k-clustering $\{A_1, A_2, ..., A_k\}$ of $X = \{x_1, x_2, ..., x_n\}$ is said to have the *String Property* iff for all x_i, x_j and x_l, and for all A_s, if $x_i, x_j \in A_s$ and $x_i < x_l < x_j$ then $x_l \in A_s$.

In the case of 1-dimensional clustering with euclidean norm ($p = 2$), it is proved that any optimal solution has the String Property [4]. In [5] this result is extended to every norm $\| \cdot \|_p$ with $p > 1$.

In this section we further extend this result to the 1-dimensional size constrained clustering problem.

First of all, we treat the case of clusterings composed by 2 clusters.

Proposition 1.5. Let $\{A, B\}$ be an optimal 2-clustering for the 2-SCC problem on instance $\{x_1, x_2, ..., x_n\}$ with $|A| = k$. Then $\{A, B\}$ has the String Property.

Proof. Consider the function $f(x) = |x - C_A|^p - |x - C_B|^p$, where C_A, C_B are the centroids of A, B respectively. By Theorem 1.4 there exists c such that $x \in A$ implies $f(x) < c$, while $x \in B$ implies $f(x) > c$. Now, suppose $C_A < C_B$. We have that:

$$\begin{aligned} \text{if } x > C_B \text{ then } f'(x) &= p((x - C_A)^{p-1} - (x - C_B)^{p-1}) > 0 \\ \text{if } C_B \geq x > C_A \text{ then } f'(x) &= p((x - C_A)^{p-1} + (C_B - x)^{p-1}) > 0 \\ \text{if } C_A \geq x \text{ then } f'(x) &= p(-(C_A - x)^{p-1} + (C_B - x)^{p-1}) > 0 \end{aligned}$$

Therefore $f(x)$ is increasing; moreover it can be easily observed that $\lim_{x \to +\infty} f(x) = +\infty$ and $\lim_{x \to -\infty} f(x) = -\infty$. Since $f(x)$ is continuous, we conclude that there is a unique x^* such that $f(x^*) = c$; moreover: $x \in A$ implies $x < x^*$, $x \in B$ implies $x > x^*$. This means that, under the assumption $C_A < C_B$, $\{A, B\}$ has the String Property. Analogous reasoning applies when $C_A > C_B$, thus yielding the String Property again. □

We notice that the two half-lines $H = \{x | f(x) < c\}$ and $\bar{H} = \{x | f(x) > c\}$ are disjoint sets; furthermore A is contained in one half-line, while B is contained in the other one. We now extend the previous result to the k-SCC.

Theorem 1.6. Let $\{A_1, A_2, ...A_k\}$ be an optimal k-clustering for SCC on instance $X = \{x_1, x_2, ..., x_n\}$ with constraints $\{m_1, m_2, ..., m_k\}$. Then $\{A_1, A_2, ..., A_k\}$ has the String Property.

Proof. Let us reason by induction on $k \geq 2$. The case $k = 2$ is clearly solved by Proposition 1.5. For $k > 2$, given an optimal k-clustering $\{A_1, A_2, ..., A_k\}$, for any j we denote $v_j = \min A_j$, $V_j = \max A_j$, and set $c = \min v_j = v_\ell$. Let us consider any index $i \neq \ell$; obviously $v_i > v_\ell$. We want to show that also $v_i > V_\ell$ holds. In fact, consider the 2-SCC problem on instance $A_\ell \cup A_i$ with constraints $\{m_\ell, m_i\}$; its optimal solution $\{A_\ell, A_i\}$ verifies the String Property because of Proposition 1.5, and hence $V_\ell \leq v_i$. As a consequence, every $A_i (i \neq \ell)$ is contained in the half-line $H = \{x | x > V_\ell\}$, while A_ℓ is contained in the complementary half-line $H^C = \{x | x \leq V_\ell\}$.
Let's now consider the optimal solution $\{A_1, ..., A_{\ell-1}, A_{\ell+1}, ..., A_k\}$ to the $(k-1)$-SCC problem on instance $X \setminus A_\ell$ with constraints $\{m_1, ..., m_{\ell-1}, m_{\ell+1}, ..., m_k\}$. By induction hypothesis, $\{A_1, ..., A_{\ell-1}, A_{\ell+1}, ..., A_k\}$ verifies the String Property, and hence by the discussion above also $\{A_1, ..., A_\ell, ..., A_k\}$ does. □

1.5 An approximation result

For integer $p > 2$, the p-centroid C of a set Y of integer numbers is an algebraic number, not necessarily rational. In this section we develop a method for obtaining an approximation of C and of $W(Y)$ by means of rational numbers. This method will be used in the following chapters for comparing two clusterings.

Given a set Y of integers $y_1 < y_2 < ... < y_m$, let j be the index such that the p-centroid C of Y verifies

$$y_j \leq C < y_{j+1}$$

Fixed ε $(0 < \varepsilon < \frac{1}{2})$, we call *$\varepsilon$-approximation* of C a number $\bar{C}$ with

$$\begin{cases} C \leq \bar{C} \leq C + \varepsilon & \text{if } y_{j+1} - C > C - y_j \\ C - \varepsilon \leq \bar{C} \leq C & \text{otherwise.} \end{cases}$$

In any case, it holds $|\bar{C} - C| \leq \varepsilon$. The intuitive idea of these technicalities means that $\bar{C}$ must be either a left or a right approximation of C in such a way as to ensure that $y_j \leq \bar{C} < y_{j+1}$.

Proposition 1.7. Given an integer $p > 2$ and m integers $1 \leq y_1 < y_2 < ... < y_m$, let C be the p-centroid of $Y = \{y_1, ..., y_m\}$ and $W(Y)$ be the cost function defined in (1.1). Then there are polynomials $A(x) = \sum_0^{p-1} a_i x^i$ and $B(x) = \sum_0^p b_i x^i$ such that:
1. C is a root of $A(x)$;
2. $W(Y) = B(C)$;
3. $|a_i|, |b_i| \leq m \cdot (y_m + 1)^p$ for every i;
4. $|B(C) - B(\bar{C})| \leq \varepsilon \cdot y_m^{p-1} \cdot p \cdot m$.

Proof. We know that there is j such that $y_j \leq C < y_{j+1}$. Consider the polynomials:

$$B(x) = \sum_{i=1}^{j} (x - y_i)^p + \sum_{i=j+1}^{m} (y_i - x)^p = \sum_0^p b_i x^i$$

$$A(x) = \frac{1}{p} B'(x) = \sum_{i=1}^{j} (x - y_i)^{p-1} - \sum_{i=j+1}^{m} (y_i - x)^{p-1} = \sum_0^{p-1} a_i x^i$$

The centroid C satisfies $A(C) = 0$; moreover $W(Y) = B(C)$. Observe now that, denoting with $[x^i]B(x)$ the coefficient of x^i in $B(x)$:

$$|b_i| = |[x^i]B(x)| \leq [x^i] \sum_{h=1}^{m} (y_h + x)^p \leq \sum_{h=1}^{m} (y_h + 1)^p \leq m(y_m + 1)^p$$

$$|a_i| \leq [x^i] \sum_{h=1}^{m} (y_h + x)^{p-1} \leq \sum_{h=1}^{m} (y_h + 1)^{p-1} \leq m(y_m + 1)^{p-1} \leq m(y_m + 1)^p$$

As for the last point we can write $|B(C) - B(\bar{C})|$ as:

$$\left|\left[\sum_{1}^{j}(C-y_i)^p+\sum_{j+1}^{m}(y_i-C)^p\right]-\left[\sum_{1}^{j}(\bar{C}-y_i)^p+\sum_{j+1}^{m}(y_i-\bar{C})^p\right]\right|=$$

$$=\left|\left[\sum_{1}^{j}(\underbrace{C-y_i}_{u})^p-(\underbrace{\bar{C}-y_i}_{\bar{u}})^p\right]+\left[\sum_{j+1}^{m}(\underbrace{y_i-C}_{-u})^p-(\underbrace{y_i-\bar{C}}_{-\bar{u}})^p\right]\right|$$

Fixed the index i in the first summation, denote $u=C-y_i$ and $\bar{u}=\bar{C}-y_i$. Since $|u|,|\bar{u}|\leq y_m$, it holds: $|u^p-\bar{u}^p|=|(u-\bar{u})(u^{p-1}+u^{p-2}\bar{u}+...+\bar{u}^{p-1})|\leq \varepsilon\cdot p\cdot y_m^{p-1}$. Observe that every single term in the last parenthesis is $u^h\bar{u}^{p-1-h}=(C-y_i)^h(\bar{C}-y_i)^{p-1-h}\leq y_m^{p-1}$, thus yielding $|u^p-\bar{u}^p|\leq \varepsilon p y_m^{p-1}$. On the other hand, when fixing the index i in the second summation, the same upper bound is obtainable. We can conclude that $|B(C)-B(\bar{C})|\leq m\varepsilon p y_m^{p-1}$. $\square$

References

1. A. Bertoni, M. Goldwurm, J. Lin, and F. Saccà. Size constrained distance clustering: separation properties and some complexity results. To appear in Fundamenta Informaticae, 2012.
2. C. Bishop. *Pattern recognition and machine learning.* Springer, 2006.
3. P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained K-Means Clustering. Technical Report MSR-TR-2000-65, Miscrosoft Research Publication, May 2000.
4. W. D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.
5. B. Novick. Norm statistics and the complexity of clustering problems. *Discrete Applied Mathematics*, 157:1831–1839, 2009.
6. 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 *Lecture Notes in Computer Science*, pages 405–419. Springer Berlin/Heidelberg, 2001.
7. H. D. Vinod. Integer Programming and the Theory of Grouping. *Journal of the American Statistical Association*, 64(326):506–519, 1969.
8. S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.

Chapter 2
Hardness

2.1 Introduction

In this chapter we show that the size constrained clustering is a difficult problem.
At this regard, fixed a norm $\| \|_p$ with $p \geq 1$, we recall the two variants of size constrained clustering problem SCC and RCC, introduced in Chapter 1.

• Size Constrained Clustering Problem (SCC): Given a point set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{Q}^d$, 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

$$|A_i| = m_i \quad \text{for } i = 1, ..., k$$

that minimizes the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

• Relaxed Constraints Clustering Problem (RCC): Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^d$, an integer $k > 1$ and a set $\mathcal{M} = \{m_1, m_2, ..., m_s\}$ of positive integers, find a k-clustering $\{A_1, ..., A_k\}$ with

$$|A_i| \in \mathcal{M} \quad \text{for all } i = 1, ..., k$$

that minimises the cost

$$W(A_1, A_2, \cdots, A_k) = \sum_1^k W(A_i).$$

Remind from Chapter 1 that we make the assumption that $X = \{x_1, ..., x_n\}$ is composed by vectors of positive integer coordinates represented in binary notation.

In the following, we will fix some of the parameters of the problems, such as the dimension d or the clustering size k (i.e. number of clusters). When the set $\mathcal{M} \subset \mathbb{N}$ is fixed (i.e. not part of the instance) we denote the problem RCC with $\mathcal{M}$-RCC.

It is simple to observe that the SCC problem with $k = 2$, is a particular case of RCC. Indeed, if in the instance of SCC the vector $(m_1, ..., m_k)$ with $\sum_1^k m_i = n$ is such that $k = 2$, then such a vector is $(m_1, n - m_1)$. The SCC problem on such an instance is equivalent to the RCC problem with constraints $\mathcal{M} = \{m_1, n - m_1\}$.
Moreover, observe that various clustering problems can be formulated as RCC. For instance the clustering problem without constraints is a particular instance of RCC where $\mathcal{M} = \{1, ..., n\}$, and the clustering problems with size inequality constraints [14, 16], e.g. $2 \leq |A_i| \leq 10$, can be formulated as RCC.

The main hardness results we obtain are:

1) For every norm $\| \, \|_p$ with $p > 1$, RCC with fixed clustering size k is NP-hard, even in the case $k = 2$ and $\mathcal{M} = \{\frac{n}{2}\}$ (Theorem 2.2).
2) For every norm $\| \, \|_p$ with $p \geq 1$, SCC with dimension d fixed is NP-hard, even in the case $d = 1$ (Theorem 2.5). Observe that RCC in dimension 1 is solvable in polynomial time [13].
3) For the euclidean norm $\| \, \|_2$, RCC in dimension $d = 2$ is NP-complete even if the possible size constraints are $\{2, 3\}$ (Theorem 2.10)

For illustrating some subtleties in the case of $\| \, \|_p$ with non-integer rational p, we consider the problem p-LC of localising the centroid of integers set $\{x_1, ..., x_n\}$ (i.e. $d = 1$).
We prove that:

4) SQRT-Sum is polynomially reducible to $\frac{3}{2}$-LC (Theorem 2.4).

This puts in evidence that it is questionable whether p-LC is in NP.

This chapter is organised as follows: in Section 2.2 we will obtain that the size constrained clustering problem with fixed k is NP-hard, in Section 2.3 we will see that when $p \geq 1$ is non-integer the problem of localising of p-centroid is in CH, in Section 2.4 we investigate the SCC when fixing the dimension d, and in Section 2.5 we will show the hardness of the relaxed version of SCC.

2.2 Constrained *k*-clustering

Let's assume in this section that the norm is $\| \, \|_p$, with $p > 1$. We will see that even fixing the number k of clusters, the solution of the size constrained k-

clustering SCC (with dimension d given in input), will be hard to determine. In particular, we consider the special case where $k = 2$ and constraints are $(\frac{n}{2}, \frac{n}{2})$. More precisely:

Definition 2.1 (Half-Partition (HP)). Given d and $X = \{x_1, ..., x_{2n}\} \subset \mathbb{N}^d$, find the optimal 2-clustering $\{A, B\}$ of X with $|A| = |B| = n$.

When $d = 1$, we call the problem HP-1. Before proceeding, we recall from Chapter 1 that in dimension $d = 1$, i.e. when $X = \{x_1, ..., x_n\}$ where $x_i \in \mathbb{R}$ for each i, the Separation Property of the optimal size constrained clustering turns out to be the so-called

String Property: a k-clustering $\{A_1, A_2, ..., A_k\}$ of $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}$ is said to have the *String Property* iff for all x_i, x_j and x_l, and for all A_s, if $x_i, x_j \in A_s$ and $x_i < x_l < x_j$ then $x_l \in A_s$.

In particular, if $k = 2$ the String Property states that there is i, $1 \leq i \leq n$, such that $\{x_1, ..., x_i\}$ is contained in a suitable half-line H, while $\{x_{i+1}, ..., x_n\}$ is contained in the complement $\bar{H}$.

HP-1 is solvable in polynomial time for any $p > 1$. Indeed, given the ordered reals $x_1 < x_2 < ... < x_{2n}$, the unique partition $\{A, B\}$ that verifies the String Property with $|A| = |B| = n$ is

$$\pi = \{\{x_1, ..., x_n\}, \{x_{n+1}, ..., x_{2n}\}\}$$

which hence turns out to be the optimal solution. Essentially, it sufficies to sort the given point set. It follows that, for any $p > 1$:

Fact 2.1. HP-1 is solvable in time $O(n \log n)$ (for any $p > 1$).

On the contrary, the problem turns to be complex when the dimension d is arbitrary; indeed we show that the HP problem is NP-hard. This implies that also 2-SCC is NP-hard.

Theorem 2.2. HP is NP-hard (for any $p > 1$).

Proof. We prove the result by a reduction from the Minimum Bisection Problem, which is known to be NP-hard [6]. This problem consists of determining, for an undirected graph $G = \langle V, E \rangle$ with $|V| = 2n$, a subset $A \subset V$ of cardinality $|A| = n$ such that the value

$$cut(A) = |\{\ell \in E \mid \ell = \{x, y\}, x \in A, y \notin A\}|$$

is minimum.

In order to construct the reduction, let $G = \langle V, E \rangle$ be an undirected graph with $V = \{1, 2, \ldots, 2n\}$ and define, for every $v \in V$, the array $X_v \in \mathbb{R}^E$ with indices in E, such that

$$X_v[\ell] = \begin{cases} 1 \text{ if } v \in \ell \\ 0 \text{ otherwise} \end{cases} \tag{2.1}$$

Thus, the family of arrays $\{X_1, X_2, \ldots, X_{2n}\}$ forms an instance of the Half-Partition problem for an arbitrary $p > 1$.

Given $A \subset V$ with $|A| = n$, let us compute its centroid C_A. For every $\ell \in E$, we have the following cases:

1. If both vertices of ℓ are in A then

$$C_A[\ell] = \arg\min_x \{2(1-x)^p + (n-2)x^p\} = \frac{1}{1 + \left(\frac{n-2}{2}\right)^{\frac{1}{p-1}}} = \alpha_n$$

2. If only one vertex of ℓ is in A then

$$C_A[\ell] = \arg\min_x \{(1-x)^p + (n-1)x^p\} = \frac{1}{1 + (n-1)^{\frac{1}{p-1}}} = \beta_n$$

3. If no vertices of ℓ is in A then $C_A[\ell] = 0$.

Now, given a 2-clustering $\{A, B\}$ with $|A| = |B| = n$, the value of objective function $W(A, B)$ can be written in the form

$$W(A,B) = \sum_{\ell \in E} \left(\sum_{i \in A} |X_i[\ell] - C_A[\ell]|^p + \sum_{j \in B} |X_j[\ell] - C_B[\ell]|^p \right)$$

If $\ell = \{i, j\}$ with $i \in A$ and $j \in B$ then we have

$$\sum_{i \in A} |X_i[\ell] - C_A[\ell]|^p + \sum_{j \in B} |X_j[\ell] - C_B[\ell]|^p = 2[(1-\beta_n)^p + (n-1)\beta_n^p]$$

On the contrary, if $\ell = \{i, j\}$ with either $\{i, j\} \subset A$ or $\{i, j\} \subset B$, then

$$\sum_{i \in A} |X_i[\ell] - C_A[\ell]|^p + \sum_{j \in B} |X_j[\ell] - C_B[\ell]|^p = 2(1-\alpha_n)^p + (n-2)\alpha_n^p$$

As a consequence, recalling that $cut(A)$ is the number of edges with a vertex in A and a vertex in B, we obtain

$$\begin{aligned} W(A,B) =& cut(A) 2[(1-\beta_n)^p + (n-1)\beta_n^p] + \\ &+ (|E| - cut(A))[2(1-\alpha_n)^p + (n-2)\alpha_n^p] \\ =& |E| \cdot g(n,p) \ + \ cut(A) \cdot s(n,p) \end{aligned} \tag{2.2}$$

where $g(n, p)$ does not depend on $\{A, B\}$ and

$$s(n,p) = 2[(1-\beta_n)^p + (n-1)\beta_n^p] - 2(1-\alpha_n)^p - (n-2)\alpha_n^p$$

Now, for any fixed $p > 1$, as n tends to $+\infty$ we have

$$\alpha_n \sim \left(\frac{2}{n}\right)^{\frac{1}{p-1}}, \quad \beta_n \sim n^{-\frac{1}{p-1}}$$

and hence

$$s(n,p) \sim (p-1)\left(2^{\frac{p}{p-1}} - 2\right) \cdot n^{-\frac{1}{p-1}} \ > \ 0$$

Therefore, from equation (2.2), if n is sufficiently large we obtain

$$\arg\min_{|A|=n} W(A,B) = \arg\min_{|A|=n} cut(A)$$

□

Corollary 2.3. The size constrained k-clustering problem with fixed k and arbitrary dimension $d > 1$ is NP-hard, for every $p > 1$.

When p is an integer, the decision version of HP is in NP. Indeed, to verify that a certain solution $\pi = \{A, B\}$ has a cost $W(\pi)$ below a given threshold $\lambda > 0$ we have to: i) formulate a system of equations with polynomials as those obtained in Proposition 1.7, and ii) use a numerical technique to compare the approximation of $W(\pi)$ with the threshold λ. In conclusion, if $p > 1$ is an integer, HP and hence 2-SCC are NP-complete.

On the other hand, when p is a non-integer rational number, this numerical approximation techniques cannot be applied and the problem seems to be not easily solvable. The next section is devoted to highlight this aspect.

2.3 Localisation of the p-centroid

When p is a non-integer rational number the p-centroid equation is not algebraic, and hence the solution seems far from trivial. To put in evidence the subtleties of this case, we briefly discuss the minor problem of localizing the p-centroid.

Definition 2.2. The problem of localizing the p-centroid (p-LC) consists of deciding, for a set X of integers $\{x_1, ..., x_n\}$ and an integer h, whether $C > h$, where C is the p-centroid of X.

It is easy to observe that the well-known SQRT-Sum problem is polinomially reducible to $\frac{3}{2}$-LC. The SQRT-Sum (or sum-of-square-roots) problem requires to decide, given positive integers $a_1, ..., a_q$, $b_1, ..., b_r$, whether $\sqrt{a_1} + ... + \sqrt{a_q} > \sqrt{b_1} + ... + \sqrt{b_r}$.

Theorem 2.4. SQRT-Sum is polinomially reducible to $\frac{3}{2}$-LC.

Proof. With the instance $a_1, ..., a_q, b_1, ..., b_r$ of SQRT-Sum we associate the instance $X = \{x_1, ..., x_{q+r}\}$ and h of $\frac{3}{2}$-LC where:

$1) h = \max a_j \qquad 2) x_i = h - a_i$ for $i \leq q \qquad 3) x_{q+j} = h + b_j$ for $1 \leq j \leq r$

Setting $F(\mu) = \sum_{i=1}^{q+r} |x_i - \mu|^{\frac{3}{2}}$, since $F(\mu)$ is strictly convex, we have:
1) $F'(\mu)$ is an increasing function;
2) if C is the $\frac{3}{2}$-centroid of X, then $F'(C) = 0$.
Observe now that

$$\frac{2}{3}F'(h) = \sum_{x_i \geq h} (x_i - h)^{\frac{1}{2}} - \sum_{x_i < h} (h - x_i)^{\frac{1}{2}} = \sqrt{b_1} + ... + \sqrt{b_r} - \sqrt{a_1} - ... - \sqrt{a_q}$$

We hence conclude that:

$$h < C \quad \text{iff} \quad F'(h) < F'(C) \quad \text{iff} \quad \sqrt{a_1} + ... + \sqrt{a_q} - \sqrt{b_1} - ... - \sqrt{b_r} > 0$$

This proves the reduction. □

The characterisation of the computational complexity of SQRT-Sum was proposed as open problem in [5]; despite the efforts, the best-known result, due to Allender et al. [1], puts SQRT-Sum in CH, i.e. the Counting Hierarchy introduced in [15], which can be defined as follows.
A language $L \subseteq \Sigma^*$ is in the class PP (abbreviation for *probabilistic polynomial-time*) iff there is a probabilistic Turing Machine (i.e. a Turing Machine equipped with the random choice operation) running in polynomial time such that, denoting with $p(w)$ the probability of accepting $w \in \Sigma^*$, it holds:

$$L = \{w \in \Sigma^* : p(w) > \frac{1}{2}\}$$

Given a family $\mathcal{L}$ of languages, a language $A \subseteq \Sigma^*$ is in the class $\mathrm{PP}^{\mathcal{L}}$ (PP relativised to $\mathcal{L}$) iff there is a probabilistic Turing Machine with oracle for $\mathcal{L}$ running in polynomial time such that $A = \{w \in \Sigma^* : p(w) > \frac{1}{2}\}$. Now, we can introduce CH in the following form [2].

Definition 2.3 (Counting Hierarchy).
- $\mathrm{C}_0 = \mathrm{P}$
- $\mathrm{C}_{i+1} = \mathrm{PP}^{C_i}$
- $\mathrm{CH} = \bigcup_i \mathrm{C}_i$

Substantially, the counting hierarchy contains: $\mathrm{C}_0 = \mathrm{P}$ (polynomial-time), $\mathrm{C}_1 = \mathrm{PP}$ (probabilistic polynomial-time), $\mathrm{C}_2 = \mathrm{PP}^{\mathrm{PP}}$ (PP relativised to PP), $\mathrm{C}_3 = \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}$ (PP relativised to $\mathrm{PP}^{\mathrm{PP}}$) and so on. It is well-known [1] that $\mathrm{NP} \subseteq \mathrm{CH} \subseteq \mathrm{PSPACE}$.

Theorem 2.4 implies that, if $\frac{3}{2}$-LC were solvable in polynomial time, then SQRT-Sum $\in$ P would hold, despite still today a major open problem is to decide whether SQRT-Sum is solvable in NP.

2.4 Constrained clustering in fixed dimension

Now we want to tackle the Size Constrained Clustering problem (SCC) and the Relaxed Constraints Clustering problem (RCC), as formulated in Section 2.1, in the case of fixed dimension d. The former problem is the subject of study in this section, while the latter one will be investigated in the following section.

We prove that the 1-dimensional size constrained clustering (SCC-1) is NP-hard, for every $p \geq 1$. First of all, we reformulate SCC-1 as a decision problem.

Definition 2.4 (SCC-1: decision version). Given a set X of n integers $x_1 < x_2 < ... < x_n$, positive integers $m_1, ..., m_k$ such that $\sum m_i = n$, and a positive integer λ (called threshold), decide whether there exists a k-clustering $\{A_1, ..., A_k\}$ of X, with constraints $|A_i| = m_i$ $(i = 1, ..., k)$, such that $W(A_1, ..., A_k) < \lambda$.

We first notice that the clustering problem without constraints is known to be solvable in polynomial time when $p = 2$. We show that adding the constraints makes the problem hard. The proof is based on a reduction from the 3-Partition problem.

Definition 2.5 (3-Partition Problem). Given a set $P = \{p_1, ..., p_{3m}\}$ of positive integers whose sum is mB, such that each p_i satisfies $B/4 < p_i < B/2$, decide whether there exists a partition $\{P_1, ..., P_m\}$ of P such that, for each $i = 1, ..., m$, $\sum_{x \in P_i} x = B$.

An equivalent version of this problem has been proved to be NP-complete in [8]; it remains NP-complete even if the numbers in P are all bounded by a polynomial in m. The problem was originally proved to be strongly NP-complete in [7] when P is a multiset.

Theorem 2.5. SCC-1 is NP-hard (for any $p \geq 1$).

Proof. We want to reduce 3-Partition to the decision version of SCC-1.
With the instance $P = \{p_1, ..., p_{3m}\}$ of 3-Partition we associate the instance of SCC-1 (decision version) given by $X = \cup_1^m X_j$ with constraints $\{p_1, ..., p_{3m}\}$ and threshold $\lambda = 3mB^{2p}$, where $X_j = \{Hj + \ell : \ell = 0, ..., B-1\}$, with $H = 6mB^2 + B$ and $B = \sum_{i=1}^{3m} p_i/m$. Now let's show the correctness of this reduction.

A partition $\{A_1, ..., A_{3m}\}$ of X is said to be *fine* if for every A_i there is X_j with $A_i \subseteq X_j$.

The main observation is that 3-Partition with instance P admits a solution if and only if there is a fine partition $\{A_1, ..., A_{3m}\}$ of X s.t. $|A_i| = p_i$ for all $i = 1, ..., 3m$. In fact, let $\{P_1, ..., P_m\}$ be a partition of P satisfying $\sum_{x \in P_i} x = B$; each P_i has 3 elements; with every $P_i = \{p_{i1}, p_{i2}, p_{i3}\}$ we associate a partition $\mathcal{A}_i = \{A_{i1}, ..., A_{i3}\}$ of X_i s.t. $|A_{ij}| = p_{ij}$ $(j = 1, 2, 3)$, which is possible since

$$\sum_{x \in P_i} x = B = |X_i|$$

Thus $\cup_1^m \mathcal{A}_i$ is a fine partition of X satisfying the constraints $\{p_1, ..., p_{3m}\}$, since $\cup P_i = P$.
Suppose now that the partition $\{A_1, ..., A_{3m}\}$ of X is fine and satisfies the constraints $|A_i| = p_i$, $i = 1, ..., 3m$. With every X_j we associate

$$P_j = \{|A_i| : A_i \subseteq X_j\}$$

Since it holds $\sum_{x \in P_j} x = |X_j| = B$, we have that $\{P_1, ..., P_m\}$ verifies the instance $\{p_1, ..., p_{3m}\}$ of 3-Partition.

To prove the correctness of the reduction, it is sufficient to observe that $\{A_1, ..., A_{3m}\}$ is a clustering of X with constraints $\{p_1, ..., p_{3m}\}$ and cost $W(A_1, ..., A_{3m}) < \lambda$ iff $\{A_1, ..., A_{3m}\}$ is fine with constraints $\{p_1, ..., p_{3m}\}$. Suppose $\{A_1, ..., A_{3m}\}$ is fine, then $W(A_1, ..., A_{3m}) = \sum_{i=1}^{3m} W(A_i)$. For all A_i there is X_j s.t. $A_i \subseteq X_j$; therefore

$$W(A_i) \leq W(X_j) < B^{p+1}.$$

In conclusion: $W(A_1, ..., A_{3m}) = \sum W(A_i) < 3mB^{p+1} \leq 3mB^{2p} = \lambda$.
Now suppose $\{A_1, ..., A_{3m}\}$ is not fine: there is A_i containing x, y with $x \in X_s$, $y \in X_t$ and $s \neq t$. Observe that $|x - y|$ is at least $H - B$; if μ is the p-centroid of A_i, then either $|x - \mu| \geq \frac{H-B}{2}$ or $|y - \mu| \geq \frac{H-B}{2}$. It follows that

$$\begin{aligned} W(A_1, ..., A_{3m}) &\geq W(A_i) \geq |x - \mu|^p + |y - \mu|^p \\ &\geq (\frac{H - B}{2})^p = (3mB^2)^p \geq 3mB^{2p} = \lambda \end{aligned}$$

□

Corollary 2.6. The size constrained clustering problem in fixed dimension d (SCC-d) is NP-hard, for any $p \geq 1$.

2.5 Relaxed Constraints Clustering

In this section we investigate the clustering problem where the cardinality of the clusters must belong to a fixed set of integers $\mathcal{M}$, as formulated in Section 2.1. Here we assume $p = 2$, i.e. we endow $\mathbb{R}^2$ with the Euclidean norm.

For $d = 1$ the problem is solvable in polynomial time for any integer $p \geq 1$ because of a dynamic programming technique found in [13].
In this section, we analyse the case $d = 2$, and we show that the relaxed constraints clustering problem in the plane is NP-hard.

More precisely, fixing a finite set $\mathcal{M} = \{m_1, ..., m_s\}$ of positive integers, let us study the following problem:

Definition 2.6 (Planar $\mathcal{M}$-RCC). Given the point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$, an integer $k > 1$, a rational $\lambda > 0$ (called threshold), decide whether there exists a clustering $\pi = \{A_1, ..., A_k\}$ of X with

$$|A_i| \in \mathcal{M} \text{ for all } i = 1, ..., k$$

such that

$$W(\pi) = W(A_1, ..., A_k) \leq \lambda.$$

It is easy to verify, from the definitions of Section 2.1, that the Planar $\mathcal{M}$-RCC problem is a particular case of the decision version of RCC.

We will to show the NP-hardness of the Planar $\mathcal{M}$-RCC problem even in the simple case $\mathcal{M} = \{2, 3\}$. As consequence, also RCC is NP-hard.
The proof is based on a technique similar to that of [11] using a reduction from the Planar 3-SAT problem, defined as follows.

Recall that a 3-CNF formula Φ is a boolean formula, written as a conjunction of clauses having exactly 3 literals. Let Φ be a 3-CNF formula with variables $V = \{v_1, ..., v_n\}$ and clauses $C = \{c_1, ..., c_m\}$; the *graph* G_Φ *of* Φ is the undirected graph $G_\Phi = \langle N, E \rangle$ with:

$$N = V \cup C \qquad E = \{\{v_i, c_j\} : v_i \text{ or } \bar{v}_i \text{ appears in } c_j\}$$

The formula Φ is said to be *planar* if its graph G_Φ is planar, i..e. admits a planar drawing (or embedding). In such a case we identify the variable $v_i \in V$ (resp. clause $c_j \in C$) with the corresponding point of the embedding in $\mathbb{R}^2$.

Definition 2.7 (Planar 3-SAT). The Planar 3-SAT problem consists in deciding, for a given planar 3-CNF formula Φ, whether there exists a satisfying assignment for Φ.

Lichtenstein [10] showed that Planar 3-SAT is strongly NP-complete. Later, Knuth and Raghunathan [9] observed 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; moreover the edges between the variables and the clauses are drawn in a rectilinear fashion [12].

Moreover, we recall some important results on planar graph drawing. An orthogonal drawing of a planar graph G is a planar embedding of G on a integer grid where each vertex is an intersection point of the grid and each edge is a chain of horizontal or vertical segments of the grid. Any planar graph with maximum degree ≤ 4 admits an orthogonal drawing. A *box-orthogonal drawing* of a planar graph G is a planar embedding of G on an integer grid where each vertex is drawn as a (possibly degenerate) rectangle of the grid and each edge is a chain of horizontal or vertical segments of the grid. Any planar graph (of arbitrary degree) admits a box-orthogonal drawing. Several algorithms have been proposed to compute the box-orthogonal drawing of a planar graph [4, 3].

Theorem 2.7 ([3]). For every planar graph $G = (V, E)$ with $|V| = n$, there is a box-orthogonal drawing for G, computable in $O(n)$ time, that uses a $a \times b$ grid, where $a + b \leq 2n$.

Before presenting the reduction from Planar 3-SAT, we furtherly need the following observation. Consider a set Y of points with rational components as shown in Figure 2.1. By inflating, Y can be drawn on an (integer) grid.

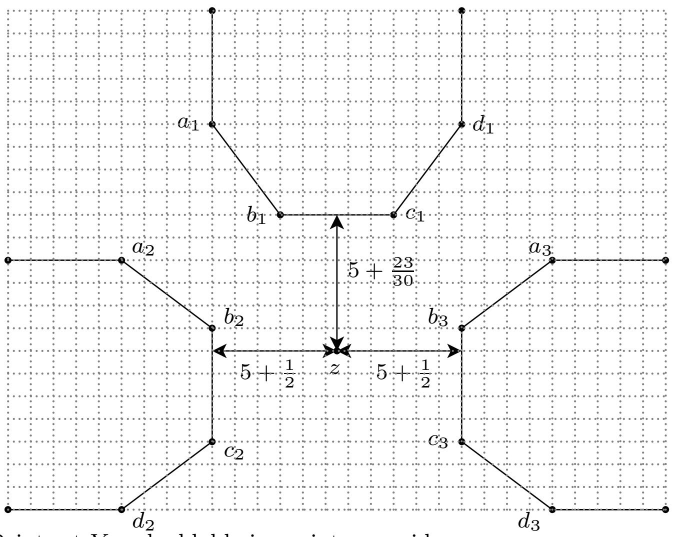

Fig. 2.1 Point set Y embeddable in an integer grid.

Lemma 2.8. It holds:

1. $W(\{z, b_1, c_1\}) = W(\{z, b_2, c_2\}) = W(\{z, b_3, c_3\}) < 35$
2. Furthermore, any other triple of points has cost > 35

Proof. Since $3^2 + 4^2 = 5^2$, the coordinates of the points a_i, b_i, c_i, d_i, $1 \leq i \leq 3$, are integers; all edges have length 5. Coordinates of z are rational.

The triangle with vertices $A = \{z, b_1, c_1\}$ is isosceles, it has base 5 and height $5 + \frac{23}{30}$; by proper rotation and translation we can easily calculate the centroid $C_A = (0, 173/90)$ and the cost

$$W(A) = \frac{23402}{675}$$

The triangle with vertices $B = \{z, b_2, c_2\}$ has base 5 and height 5.5; by proper rotation and translation we can easily calculate the centroid $C_B =$

$(173/90, 11/6)$ and the cost $W(B) = \frac{23402}{675} = W(A) < 35$. The triangle with vertices $C = \{z, b_3, c_3\}$ is symmetric to A, hence has the same cost.

Consider the triangle with vertices $A' = \{z, b_1, b_2\}$; after placing the origin at b_2 we can easily check that the centroid is $C_{A'} = (17/6, 173/90)$ and the cost is $W(A') = \frac{8002}{225} > 35$. Consider the triangle with vertices $B' = \{a_2, b_2, b_1\}$; by proper rotation and placing the origin at a_2 we can easily calculate the centroid $C_{B'} = (7/3, 0)$ and the cost $W(B') = 128/3 > 35$.

It is evident that other sets of 3 vertices has cost > 35. □

Remark 2.1. We remark that by properly rescaling the grid by a factor $\frac{1}{30}$, also the point z can be embedded to a intersection point of the grid.

We are ready now to illustrate the reduction.

Theorem 2.9. Planar 3-SAT is polynomially reducible to Planar $\{2,3\}$-RCC.

Proof. Let Φ be the instance of Planar 3-SAT with n variables $V = \{v_1, ..., v_n\}$ and m clauses $C = \{c_1, ..., c_m\}$.
The basic idea is to construct a proper planar graph G from the instance Φ such that, the cost of the optimal clustering on the points of G with m clusters of size 2 or 3 is less than or equal to a suitable threshold $\lambda > 0$ if and only if the formula Φ is satisfiable.

With the instance Φ we associate a point set X in a box-orthogonal grid, as follows.

1. For every clause $c_j \in C$ we associate a point z_j in the plane.
2. For every variable $v_i \in V$ we associate a simple circuit Γ_i in the plane containing $2L_i$ consecutive points $x_{i1}, ..., x_{i(2L_i)}$. x_{it} and $x_{i(t+1)}$ are at distance 5, for every t $(1 \le t \le 2L_i - 1)$, and so are $x_{i(2L_i)}$ and x_{i1}, while $\|x_{it} - x_{is}\| > 5$ for $1 < |t - s| < 2L_i - 1$.
3. For every clause c_j and every variable v_i appearing in c_j, there are two consecutive points x_{it} and $x_{i(t+1)}$ which are the nearest points to z_j among the points of the circuit Γ_i: $x_{i1}, ..., x_{i(2L_i)}$. In such a case we say that c_j (or z_j) *touches* the cluster $\{x_{it}, x_{i(t+1)}\}$. If the clause c_j contains the literal x_i then t is even; if c_j contains the literal $\overline{x_i}$ then t is odd.
4. The segment between the points $\{x_{it}, x_{i(t+1)}\}$ touched by a clause c_j can be horizontal or vertical. If it is horizontal, its distance from z_j is $5 + \frac{23}{30}$, otherwise $5 + 0.5$. More precisely, the arrangement of points in a neighborhood of z_j is given by the Figure 2.2.

By Theorem 2.7 such a grid can can be obtained in time $O(2\sum_1^n L_i + m)$ and the rescaling factor can be chosen properly to draw the z_j's as intersection points.

We observe that, for every variable v_i, the points $x_{i1}, ..., x_{i(2L_i)}$ of Γ_i admit only two optimal clustering with clusters of size 2:

$$\pi_1 = \{\{x_{i1}, x_{i2}\}, \{x_{i3}, x_{i4}\}, ...\} \qquad \pi_1 = \{\{x_{i(2L_i)}, x_{i1}\}, \{x_{i2}, x_{i3}\}, ...\}$$

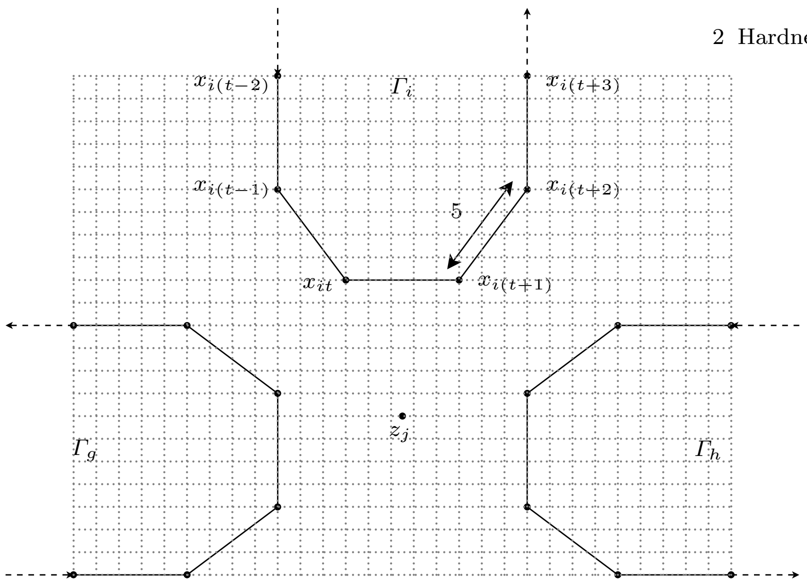

Fig. 2.2 The neighborhood of a point z_j associated to clause c_j contains points from the 3 circuits associated to the variables appearing in c_j.

The clustering π_1 corresponds to the True assignment to the variable x_i, while π_2 corresponds to the False assignment to x_i, so that: if x_i is a literal in c_j then c_j touches a cluster in π_1, if $\overline{x_i}$ is a literal in c_j then c_j touches a cluster in π_2.

Moreover, we notice that every cluster $\{x_{is}, x_{i(s+1)}\}$, has cost $5^2/2$, while if c_j touches $\{x_{it}, x_{i(t+1)}\}$ the cluster $\{z_j, x_{it}, x_{i(t+1)}\}$, has cost 23402/675 as stated by Lemma 2.8.

We observe that each assignment to $(x_1, ..., x_n)$ select, for every variable x_i, one of the two optimal clustering of the points in Γ_i. Hence, an assignment satisfies Φ if and only if every clause c_j touches at least one cluster of a selected clustering.

Now, we construct the instance (X, k, λ) of the $\{2, 3\}$-RCC problem associated to Φ:

1. $X = \{z_1, ..., z_m\} \cup \{x_{is} : i = 1, ..., n; s = 1, ..., 2L_i\}$
2. $k = \sum_1^n L_i$
3. $\lambda = mT + (k - m)\frac{5^2}{2}$, where $T = \frac{23402}{675}$

Every clustering with size constraints $\{2, 3\}$, whose clusters we call *segments* and *triangles*, must contain exactly m triangles. Indeed, if n_T is the number of triangles and n_S is the number of segments in a clustering, the following equations hold:

$$2n_S + 3n_T = 2\sum_{1}^{n} L_i + m \qquad \text{(total number of points)}$$

$$n_S + n_T = k \qquad \text{(total number of clusters)}$$

which yields $n_T = m$.

Since the triangles have minimum cost T by Lemma 2.8, and the segments have minimum cost $5^2/2$, the cost of every clustering π is:

$$W(\pi) \geq mT + (k-m)\frac{5^2}{2}$$

Now, to complete the reduction we verify that the cost of the solution of the instance (X, k, λ) is λ iff Φ is satisfiable. Suppose Φ is satisfiable, that is there exists an assignment which satisfies it. For such an assignment, every clause c_j touches at least one pair $\{x_{it}, x_{i(t+1)}\}$ of the selected clustering and hence an optimal triangle $\{x_{it}, x_{i(t+1)}, z_j\}$. The cost of the obtained clustering is exactly

$$mT + (k-m)\frac{5^2}{2} = \lambda$$

Vice versa, if there exists a clustering with cost λ, such a clustering must contain m minimal cost triangles. The only triangles with minimum cost are those containing one of the points $\{z_1, ..., z_m\}$; two such points cannot lie in the same cluster, hence every point z_j touches some pair of a selected clustering and therefore the formula Φ is satisfiable. □

Since Planar 3-SAT is NP-complete, the $\{2,3\}$-RCC problem in the plane is NP-hard. Since $\{2,3\}$-RCC is trivially in NP, we conclude:

Theorem 2.10. The $\{2,3\}$-RCC problem in the plane is NP-complete.

References

1. E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. In *Proc. 21st Ann. IEEE Conf. on Computational Complexity (CCC'06)*, pages 331–339, 2006.
2. E. Allender and K. W. Wagner. Counting Hierarchies: Polynomial Time and Constant. *Bulletin of the EATCS*, 40:182–194, 1990.
3. U. Fößmeier, G. Kant, and M. Kaufmann. 2-Visibility drawings of planar graphs. In S. North, editor, *Proc. Symposium on Graph Drawing, GD'96*, volume 1190 of *Lecture Notes in Computer Science*, pages 155–168. Springer Berlin, 1997.
4. U. Fößmeier and M. Kaufmann. Drawing High Degree Graphs with Low Bend Numbers. In F. Brandenburg, editor, *Proc. Symposium on Graph Drawing, GD'95*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266. Springer Berlin, 1996.
5. M. Garey, R. Graham, and D. Johnson. Some NP-complete geometric problems. In *Proceedings of the eighth annual ACM symposium on Theory of Computing*, pages 10–22, NY, USA, 1976. ACM New York.
6. M. R. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 1979.
8. H. Hulett, T. G. Will, and G. J. Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594 – 596, 2008.
9. D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.
10. D. Lichtenstein. Planar Formulae and Their Uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
11. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The Planar k-Means Problem is NP-Hard. In S. Das and R. Uehara, editors, *WALCOM: Algorithms and Computation*, volume 5431 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin/Heidelberg, 2009.
12. W. Mulzer and G. Rote. Minimum-weight triangulation is np-hard. *J. ACM*, 55(2), 2008.
13. F. Saccà. *Problemi di Clustering con Vincoli: Algorithmi e Complessità*. PhD thesis, University of Milan, Milan, 2010.
14. 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 *Lecture Notes in Computer Science*, pages 405–419. Springer Berlin/Heidelberg, 2001.
15. K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.
16. S. Zhu, D. Wang, and T. Li. Data clustering with size constraints. *Knowledge-Based Systems*, 23(8):883–889, 2010.

Chapter 3
Clustering in the Plane with Euclidean Norm

3.1 Introduction

We proved in the previous chapter that the problem $\{2,3\}$-RCC in the plane is NP-complete, so we can't expect to obtain efficient exact algorithms for the general problem RCC in the plane.

In this chapter we fix the norm $\| \|_2$ and we investigate RCC in the plane, with a fixed clustering size s (i.e. the number of clusters is not part of the instance). In particular we study the case $s = 2$.

We consider the following problem:

Definition 3.1 (2-RCC in the Plane (briefly 2-RCC)). Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$ and $\mathcal{M} = \{k, n-k\}$, find a 2-clustering $\{A, \bar{A}\}$ of X with $|A| = k, |\bar{A}| = n-k$, that minimises

$$W(A, \bar{A}) = W(A) + W(\bar{A})$$

We also consider the following:

Definition 3.2 (Full 2-RCC in the plane (briefly Full 2-RCC)). Given a point set $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$, find all the optimal 2-clusterings $\pi_k = \{A_k, \bar{A}_k\}$, with $|A_k| = k$ for all k, $1 \leq k \leq \lfloor \frac{|X|}{2} \rfloor$.

The main results we obtain are:

1) There is an algorithm for solving Full 2-RCC problem in time $O(n^2 \cdot \log n)$ (Theorem 3.2).
2) There is an algorithm for solving 2-RCC problem in time $O(n\sqrt[3]{k} \cdot \log^2 n)$ (Theorem 3.17).

The algorithm for solving Full 2-RCC is designed and analysed in Section 3.2. The algorithm for solving 2-RCC is more difficult, and it requires methods related to the challange-problem of Combinatorial Geometry of enumerating the k-sets of points X in the plane.

In Section 3.3 such methods are reviewed, while in Section 3.4 the algorithm is described at a high abstraction level (Algorithm 3).
The implementation of Algorithm 3 requires an efficient dynamical data structure for constructing and maintaining the convex hull of a planar point set, under the operations of insertion, deletion and query. This implementation is described in Section 3.5.

3.2 Size Constrained 2-Clusterings in the Plane

In this section we present an algorithm that, taking in input a finite subset X of points of the plane, outputs all the optimal 2-clusterings $\{A_k, \bar{A}_k\}$ of X with constraint $|A_k| = k$, for all k, $1 \leq k \leq \lfloor |X|/2 \rfloor$. This algorithm works in time $O(n^2 \log n)$. In this chapter we simplify the notation by writing $\| \ \|$ for the norm $\| \ \|_2$.

We suppose that $X = \{x_1, ..., x_n\} \subset \mathbb{Q}^2$ is in *general position*, i.e. no triple (x, y, z) of distinct points of X is collinear. Moreover, for sake of simplicity we impose two hypotheses on the point set X:

(a) $\sum_{x \in X} x = 0$
(b) the points of X have all distinct ordinates

The hypothesis (a) can be assumed w.l.o.g., since a translation of X does not change the clustering costs, while the hypothesis (b) can be overcome with minor modifications (e.g. by slight rotation of X).

We recall that, for the norm $\| \ \|_2$, the centroid μ_A of a point set A is the component-wise mean:

$$\mu_A = \left(\sum_{x \in A} x \right) / |A| \tag{3.1}$$

Given a clustering $\pi = \{A, \bar{A}\}$ of X, because of hypothesis (a) and eq. (3.1) we have that $|A|\mu_A + |\bar{A}|\mu_{\bar{A}} = 0$. Moreover, the cost $W(A, \bar{A})$ of the clustering π can be obtained as follows:

$$\begin{aligned} W(A, \bar{A}) &= \sum_{x \in A} \|x - \mu_A\|^2 + \sum_{x \in \bar{A}} \|x - \mu_{\bar{A}}\|^2 \\ &= \sum_{x \in X} \|x\|^2 - |A| \|\mu_A\|^2 - (n - |A|) \|\mu_{\bar{A}}\|^2 \end{aligned} \tag{3.2}$$

If we set

$$S_A = \sum_{x \in A} x$$

it follows that

$$S_{\bar{A}} = -S_A \tag{3.3}$$

$$W(A, \bar{A}) = \sum_{x \in X} \|x\|^2 - \frac{\|S_A\|^2 n}{|A|(n - |A|)} = \sum_{x \in X} \|x\|^2 - \frac{\|S_{\bar{A}}\|^2 n}{|\bar{A}|(n - |\bar{A}|)} \tag{3.4}$$

Because of (3.4), the optimal size constrained 2-clustering $\pi = \{A, B\}$ with $|A| = k$ can be obtained as follows:

$$\operatorname*{arg\,min}_{A \subset X : |A| = k} W(A, \bar{A}) = \operatorname*{arg\,max}_{A \subset X : |A| = k} \frac{\|S_A\|^2}{k(n-k)} \tag{3.5}$$

Definition 3.3. We say that $A \subset X$ is a *k-set* of X if $|A| = k$ and $A = X \cap H$ for a suitable half-space $H \subset \mathbb{R}^d$.

In other words, a subset A is a k-set if it is separated from $\bar{A}$ by a straight line ℓ and has k points. Now we recall from Section 1.3 that an optimal clustering $\pi = \{A, \bar{A}\}$ with $|A| = k$ is such that there is a straight line ℓ separating A form $\bar{A}$. We can conclude the following

Fact 3.1. The problem of finding the optimal 2-clustering $\pi = \{A, \bar{A}\}$ with $|A| = k$ is reduced to the problem:

$$\operatorname*{arg\,min}_{A \subset X : |A| = k} W(A, \bar{A}) = \operatorname*{arg\,max}_{A \text{ is } k\text{-set of } X} \frac{\|S_A\|^2}{k(n-k)}$$

In order to solve the problem

$$\operatorname*{arg\,max}_{A \text{ is } k\text{-set of } X} \frac{\|S_A\|^2}{k(n-k)}$$

we observe that, if A is separated from $\bar{A}$ by an oriented straight line ℓ that keeps A on its right-hand side, then there is a pair (x, y) of distinct points in X such that

i) $y \in A$, $x \in \bar{A}$
ii) the straight line through x and y separates $A \setminus \{y\}$ from $\bar{A} \setminus \{x\}$

Vice versa, with a pair (x, y) of distinct elements in X, we can associate the ordered bipartition $(A, \bar{A})$ of X s.t. A is the set of points on the right of the oriented straight line (x, y), including y and excluding x. We conclude that the ordered bipartitions $(A, \bar{A})$, where A is separable from $\bar{A}$ by a straight line ℓ, are in bijection with the set of oriented edges

$$E = \{(x, y) : x, y \in X, x \neq y\}$$

Remark 3.1. We observe that the optimal clustering of n points with constraint k is exactly the optimal clustering with constraint $n-k$. It is then sufficient to find all the optimal clustering for $1 \leq k \leq \lfloor n/2 \rfloor$.

These facts suggest the following Algorithm 1, written at a very high level of abstraction, for finding all the optimal 2-clusterings $\pi_k = \{A_k, \bar{A}_k\}$ with $|A_k| = k$, for all k, $1 \leq k \leq \lfloor |X|/2 \rfloor$.

Algorithm 1 Full 2-RCC

```
Input: a point set $X = \{x_1, ..., x_n\} \subset \mathbb{R}^2$ in general position, having distinct ordinates, s.t. $\sum_{x \in X} x = 0$
Output: optimal clusterings $\pi_k = \{A_k, \overline{A_k}\}$ with $|A_k| = k$, $1 \leq k \leq \lfloor n/2 \rfloor$
1  for $1 \leq k \leq \lfloor n/2 \rfloor$ do
2        $q_k = 0$
3  for all pairs $(x_i, x_j)$, $(i \neq j)$, do
4        $A :=$ set of points in $X$ on the right of the oriented straight line $(x_i, x_j)$
5        $S_A = \sum_{x \in A} x$
6        $g = |A|$
7        $m = \min\{g, n-g\}$
8        if $m \neq 0$ and $\frac{\|S\|^2}{g(n-g)} > q[m]$ then
9              $\pi_m = \{A, \bar{A}\}$;
10             $q[m] = \frac{\|S\|^2}{g(n-g)}$
11 return $(\pi_1, ..., \pi_{\lfloor n/2 \rfloor})$
```

We notice that the number of iterations of the for-loop at line 3 is $O(n^2)$. To obtain a near $O(n^2)$ time algorithm it is then desirable to design an efficient method for executing each iteration. To this end, the basic idea inspired by the work of Hansen et al. [13] is to enumerate the pairs (x, y) in a suitable manner to allow an immediate calculation of the new value of the program variable S_A at line 5 from its old value at the previous step. In fact, for a fixed x the enumeration will be done, in the order of the angular coefficient of the straight line through x and y.

In details, let $X = \{x_1, ..., x_n\}$ be the set of n points in $\mathbb{R}^2$ in general position and satisfying the hypotheses (a) and (b). Consider a point $x \in X$, and let y be another point of X. We denote by $e = (x, y)$ the oriented edge from x to y, and with little abuse of language we also denote by (x, y) the straight line through x and y. We define the *angle* $\measuredangle x$ of a vector $x \in \mathbb{R}^2$ as the oriented angle between $(1, 0)$ and x in counterclock-wise sense. The *slope of a vector* $x \in \mathbb{R}^2$ is $\langle \measuredangle x \rangle_\pi \in [0, \pi)$, i.e. the remainder of the division of $\measuredangle x$ by the constant π; while the *slope of an edge* $e = (x, y)$ is the slope of $(y - x)$. By this definition, the slope of x is equal to the slope of $-x$ and the slope of (x, y) is equal to the slope of (y, x).

We will introduce a total order $<$ on the edge set $E = \{(x_i, x_j) : i \neq j\}$. First of all, sort $x_1, ..., x_n$ by ordinate, so that w.l.o.g. we can suppose

$x_1 < x_2 < ... < x_n$. This can be easily achieved in time $O(n \log n)$. Fixed i $(1 \leq i \leq n)$, define the set of oriented edges from x_i to the other points (usually called forward star):

$$FS(x_i) = \{(x_i, x_j) : x_j \in X, i \neq j\}.$$

and sort the set $FS(x_i)$ by slope, thus obtaining:

$$(x_i, y_{i1}) < (x_i, y_{i2}) < ... < (x_i, y_{i(n-1)})$$

Then the total order $<$ on E can be given by:

$$(x_i, y_{ij}) < (x_k, y_{ks}) \qquad \Longleftrightarrow \qquad i < k \text{ or } i = k \text{ and } j < s.$$

This ordering can be achieved in $O(n \log n)$ time since it takes constant time to calculate the slope of an edge (x_i, x_j).

Now, for every $x_i \in X$ consider the horizontal line through x_i: it determines the partition $\pi_{i0} = \{A_{i0}, \overline{A_{i0}}\}$ with

$$A_{i0} = \{y \in X : y \text{ is below the straight line through } x_i\}$$

which is obviously a $|A_{i0}|$-set. That can be done in $O(n)$ time. At the end, for every (x_i, y_{ij}), let

$$A_{ij} = \{z \in X : z \text{ lies on the right of the straight line } (x_i, y_{ij})\}$$

It is easy to observe that $A_{i(j+1)}$ can be easily obtained from A_{ij} by means of:

$$A_{i(j+1)} = \begin{cases} A_{ij} \cup \{y_{ij}\} & \text{if } y_{ij} - x_i \text{ has positive ordinate} \\ A_{ij} \setminus \{y_{ij}\} & \text{otherwise} \end{cases}$$

which is clearly a $|A_{i(j+1)}|$-set and hence determines the clustering $\pi_{i(j+1)} = \{A_{i(j+1)}, \overline{A_{i(j+1)}}\}$, which differs from π_{ij} by a so-called switch of the point y_{ij} from one cluster to the other.

The enumeration of the edges (x_i, y_{ij}) and the associated partitions π_{ij} can be interpreted as a rotating straight line with center x_i in counterclockwise sense. When the rotating line crosses the point y_{ij}, if y_{ij} was on the right (resp. left) of the rotating line it passes to the left (resp. right), thus yielding two new clusters $A_{i(j+1)}$ and $\overline{A_{i(j+1)}}$.

Denoting $S_{ij} = S_{A_{i(j+1)}}$ and $g_{ij} = |A_{ij}|$, we observe that the value of $S_{i(j+1)}$ and $g_{i(j+1)}$ can be easily obtained by the recurrences:

$$S_{i(j+1)} = \begin{cases} S_{ij} + y_{ij} & \text{if } y_{ij} - x_i \text{ has positive ordinate} \\ S_{ij} - y_{ij} & \text{otherwise} \end{cases} \tag{3.6}$$

$$g_{i(j+1)} = \begin{cases} g_{ij} + 1 & \text{if } y_{ij} - x_i \text{ has positive ordinate} \\ g_{ij} - 1 & \text{otherwise} \end{cases} \tag{3.7}$$

It is also interesting to note that the centroid of A_{ij} and its complement is obtainable, respectively, as:

$$\mu_{A_{ij}} = \frac{S_{ij}}{g_{ij}} \qquad \mu_{\overline{A_{ij}}} = \frac{-S_{ij}}{(n - g_{ij})}$$

To enumerate all the possible 2-clusterings of X, it suffices to repeat the considerations above with all the other points $x_i \in X$. We formalise these steps in the Algorithm 2. The complexity analysis was almost presented.

Algorithm 2 Full 2-RCC

```
Input: a point set $X = \{x_1, ..., x_n\} \subset \mathbb{R}^2$ in general position, having distinct ordinates, s.t. $\sum_{x \in X} x = 0$
Output: sequence $(e[1], ..., e[\lfloor n/2 \rfloor])$ of edges where $e[k]$ is associated with the solution of 2-RCC with constraints $\mathcal{M} = \{k, n-k\}$
1  for $1 \leq k \leq \lfloor n/2 \rfloor$ do
2      $q[k] = 0$
3  for $i = 1, ..., n$ do
4      $A =$ points of $X$ below the horizontal line through $x_i$
5      $S = \sum_{x \in A} x$
6      $g = |A|$          // it can be $g = 0$
7      Sort $((x_i, x_1), (x_i, x_{i-1}), (x_i, x_{i+1}), ..., (x_i, x_{n-1}))$ by slope obtaining $((x_i, y_{i1}), ..., (x_i, y_{i(n-1)}))$
8      for $j = 1, ..., n-1$ do
9          if $y_{ij}$'s ordinate $> x_i$'s ordinate then
10             $S = S + y_{ij}$; $g = g + 1$
11         else
12             $S = S - y_{ij}$; $g = g - 1$
13         $m = \min\{g, n-g\}$
14         if $m \neq 0$ and $q[m] < \frac{\|S\|^2}{g(n-g)}$ then
15             $q[m] = \frac{\|S^2\|}{g(n-g)}$
16             $e[m] = (x_i, y_{ij})$
17 return $(e[1], ..., e[\lfloor n/2 \rfloor]))$
```

It suffices to note that the for-loop at line 3 iterates n times. Lines 4–6 cost $O(n)$ steps, while the heaviest calculation is the sorting at line 7 which requires $O(n \log n)$. The operations in lines 9–16 require constant time and are repeated $n-1$ times. In conclusion the computational time is $T(n) = n[O(n) + O(n \log n) + (n-1)O(1)] = O(n^2 \log n)$.

Theorem 3.2. Let X be a set of n points in the plane in general position having distinct ordinates and satistying $\sum_{x \in X} = 0$. There is an algorithm that finds all the optimal 2-clusterings of X with every constraint k, $1 \leq k \leq \lfloor n/2 \rfloor$ running in time $O(n^2 \log n)$.

We underline that all the 2-SCC-2 problems with constraints k, $1 \leq k \leq \lfloor n/2 \rfloor$, are solved with one execution of the Algorithm 2. In the next sections we will see instead that, if we want to compute only one optimal k-clustering for a given k, the time complexity can be furtherly reduced by geometrical technique.

Since such a technique is based on an efficient method for iterating through all possible k-sets, with fixed k, in the next section we recall some methods of combinatorial geometry that allows one to estimate the number of k-sets in the plane.

3.3 k-Sets

This section is devoted to introduce some notions and results of *combinatorial geometry*, which is the study of combinatorial or discrete properties of geometrical objects in finite-dimensional spaces, such as points and lines. Moreover, when the interest switches toward algorithmic techniques to solve combinatorial geometry problems the study enters into the field of computational geometry. The branch of combinatorial geometry was initially established through various works by classical mathematicians such as Euler and Kepler, and more recently by modern authors such as H. Minkowski, L. F. Tóth and the prolific mathematician P. Erdös.

Two central topics in combinatorial geometry are the study of separation of a point set and the arrangement of hyperplanes. Among the former topic an old problem, first studied by Lovász [16] and Erdös [12], consists in determining the number of possible separations of a given point set P in the plane obtainable by lines. The problem can be extended to dimension $d > 2$ using the hyperplanes. It is well-known that this extended problem is equivalent to asking how many cells the space is cut into by a suitable set of hyperplanes, obtained by a duality construction from the point set P [9].

Now, we will formalise the above-mentioned notions and illustrate the main tool that consists in the bound on the number of planar k-sets. Let $X = \{x_1, ..., x_n\}$ be a set of n points in $\mathbb{R}^d$. We will say that they are in *general position* if any $d+1$ points taken from X are not coplanar points. In the Euclidean space $\mathbb{R}^3$ this condition can be easily checked by the equality

$$(y_3 - y_1) \cdot [(y_2 - y_1) \times (y_4 - y_3)] = 0$$

for any 4 distinct points $y_i \in X$, $i = 1, ..., 4$ ($\cdot$ and $\times$ denote inner and cross product respectively). Unless otherwise stated, we will always work with this assumption, since it doesn't lead to a loss of generality [9]. Recall the following definition already introduced in Section 3.2.

Definition 3.4. Given a finite set X of n points in $\mathbb{R}^d$ and an integer $0 \leq k \leq n$, a *k-set* is a subset $A \subseteq X$ with cardinality $|A| = k$ such that $A = X \cap H$ for a suitable half-space $H \subset \mathbb{R}^d$.

When $d = 2$, the k-sets are those planar point sets A separated from $X \setminus A$ by a straight line and are thus called *planar k-sets*. Denote by $f_k^{(d)}(X)$ the number of k-sets of $X \subset \mathbb{R}^d$ and by $f_k^{(d)}(n)$ the maximum number of k-sets for n arbitrary points in $\mathbb{R}^d$:

$$f_k^{(d)}(n) = \max_{X \subset \mathbb{R}^d, |X|=n} f_k^{(d)}(X)$$

Moreover, we will write $f_k(n)$ when $d = 2$, i.e. $f_k(n) = f_k^{(2)}(n)$.

Remark 3.2. We notice that $f_k(n) = f_{n-k}(n)$.

Definition 3.5 (k-Sets Problem). Fixing the dimension d, and given n and k, determine the number $f_k^{(d)}(n)$. In the case $d = 2$, the problem of determining $f_k(n)$ is called Planar k-Sets Problem.

We want to recall both some lower bound and upper bound on $f_k(n)$. A special instance of the problem where $n = 2k$ is called *Halving Planes Problem*. Furthermore, the particular case of the Halving Planes Problem where $d = 2$ is called *Halving Lines Problem*, and was studied initially in 1971 by Lovász [16], where an effective method was given for constructing point sets X with the lower bound $f_{n/2}(n) = \Omega(n \log n)$. A few years later, Erdös et al. [12] discovered an extension of this result for the Planar k-Sets Problem, giving a lower bound $f_k(n) = \Omega(n \log k)$; the same result was found independently by Edelsbrunner and Welzl [10]. After long time with no actual developments other than small improvements on the constant of the lower bound, a little step was done by Tóth [26] demonstrating that $f_{n/2}(n) = ne^{\Omega(\sqrt{\log n})}$ and $f_k(n) = ne^{\Omega(\sqrt{\log k})}$, and improving a similar bound for the complexity of the median level in pseudoline arrangements, which is the dual equivalent [9] of the Halving Lines Problem, although the manuscript by Klawe et al. containing this bound remained unpublished [14].

On the other hand, the existing upper bounds for the Planar k-Sets Problem seem to be far larger than the best-known lower bound. By improving the $O(n^{3/2})$ upper bound of Lovász [16] for the Halving Lines Problem, Erdös et al. [12] in 1973 generalised the result to an upper bound of $f_k(n) = O(n\sqrt{k})$ for the Planar k-Sets Problem. After the result by Erdös et al. tiny improvements were done in literature, such as that one by Pach et al. [18, 19] consisting in $f_k(n) = O(n\sqrt{k}/\log^* k)$ ($\log^*$ being the iterated logarithm), until the significant step 26 years later due to Dey [8] which gives a new upper bound $f_k(n) = O(n\sqrt[3]{k})$.

It is interesting to recall that there are extension of these results for higher-dimensional spaces. In particular, in the Euclidean space $\mathbb{R}^3$ the k-Sets Problem has an upper bound $f_k^{(3)}(n) = O(nk^{3/2})$ due to Sharir et al. [25] and a lower bound $f_k^{(3)}(n) = nke^{\Omega(\sqrt{\log k})}$ [26], which easily generalises to $f_k^{(d)} = nk^{d-1}e^{\Omega(\sqrt{\log k})}$ for d-dimensional spaces.

Although the difficulties to find a strict upper bound for the Planar k-Sets Problem, there's an exact evaluation of the maximum number of $\leq k$-Sets in the plane, i.e. the number $g_k(n) = \sum_{i \leq k} f_i(n)$, for arbitrary n points; indeed Alon and Györi [3] showed a combinatorial proof based on n-sequences that leads to $g_k(n) = nk$.

In this section we want to illustrate the result by Dey [8] in 1998.

Remark 3.3. Before this presentation, it is noteworthy to give a brief survey on a equivalent result in the dual space [9] that was proved by the same author in the previous year [7]. Intuitively, the *geometric duality* is a one-to-one transformation $\mathcal{D}$ that maps points of $\mathbb{R}^d$ to non-vertical hyperplanes of $\mathbb{R}^d$ and vice versa. Formally, the dual of the point $p = (a_1, ..., a_d)$ is the hyperplane $p^* = \mathcal{D}(p)$:

$$p^* : x_d = a_1 x_1 + ... + a_{d-1} x_{d-1} - a_d$$

The dual of the hyperplane $h : x_d = a_1 x_1 + ... + a_{d-1} x_{d-1} + a_d$ is the point $h^* = \mathcal{D}(h) = (a_1, ..., a_{d-1}, -a_d)$. In particular, in the Euclidean plane a point $p = (a, b)$ is mapped to the line $p^* : y = ax - b$, and the line $\ell : y = ax + b$ is mapped to the point $\ell^* = (a, -b)$. There exists also an alternative definition of geometric duality, called polar duality [5]. Independently of which definition is taken, the duality is an involution, i.e. $(p^*)^* = p$, $(\ell^*)^* = \ell$, and preserves the order and incidence relations between point and lines, so that many theorems in geometry has an equivalent dual formulation. A *k-level* in the arrangement of n lines is the closure of the set of points p, with p lying on one line and having exactly k lines strictly below it. The *complexity* of the k-level is the number of its vertices. It is well-known that determining the complexity of the k-level in the arrangement of the dual lines of a given set X of n points is equivalent to the Planar k-Sets Problem. By exploiting an old result [2, 15] on the crossing number of a graph, i.e. the minimum number of edge intersections of a planar drawing of the graph, Dey [7] succeded in demonstrating the $O(n\sqrt[3]{k})$ upper bound for the complexity of the k-level in the arrangement of n lines and hence for the maximum number of k-sets of n points.

Now, we give a short survey illustrating the upper bound [8] for $f_k(n)$ in the primal space, i.e. in the space $\mathbb{R}^2$ containing the point set X. The proof is based on some preliminary results on the so-called convex chains and exploits again the crossing number inequality in [2, 15].
Let's consider the set X of n points in general position in the Euclidean plane $\mathbb{R}^2$. Given a pair of points $p, q \in X$ we denote by ℓ_{pq} the straight line through p and q, and by $\overline{pq}$ the line segment having p and q as endpoints; moreover we say that ℓ_{pq} is the *supporting line* of $\overline{pq}$.

Definition 3.6. The line segment $\overline{pq}$ is called *k-set edge* of X if there are exactly k points of X in one of the two open half-plane determined by the supporting line ℓ_{pq}.

In other words, $\overline{pq}$ is a k-set edge if:

$$|X \cap H| = k \quad \text{with } H \text{ open half-plane having boundary } \partial H = \ell_{pq}$$

We can decide to give an orientation to a k-set edge $\overline{pq}$ by denoting it with (p, q) or (q, p). Let $e = (p, q)$ be an oriented k-set edge; we define N_e^+ as the number of points on the left open half-plane w.r.t. e. We can then define the edge set $E_k = \{e = (p, q) \in V \times V : N_e^+ = k\}$.

Construct a geometric graph $G_k = \langle V, \overrightarrow{E}_k \rangle$ in the following manner. Let $V = X$. Let $\overrightarrow{E}_k$ be the set of k-set edges of X oriented from left to right (we assume there are no vertical k-set edge, otherwise it suffices a small rotation of X) having exactly k points of X on its left open half-plane. i.e. we write

$$\overrightarrow{E}_k = \{e = (p, q) \in V \times V : p_y < q_y, N_e^+ = k\}.$$

Clearly $\overrightarrow{E}_k \subseteq E_k$. We can assume without loss of generality that the remaining oriented k-set edges not in $\overrightarrow{E}_k$ are less than those in $\overrightarrow{E}_k$, i.e. the number of k-set edges is $|E_k| < 2|\overrightarrow{E}_k|$. We call such a graph G_k the (directed) *k-graph* of X. Given an edge $e \in \overrightarrow{E}_k$ we denote by L_e and R_e the left and right endpoint respectively, and by S_e the slope of the supporting line ℓ_e of e (or simply the slope of e).

Proposition 3.3 ([12]). Let $p \in V$ and let $a, b \in \overrightarrow{E}_k$ be two incoming (resp. outgoing) edges having slopes $S_a < S_b$; then there exists an outgoing (resp. incoming) edge $e \in \overrightarrow{E}_k$ having slope $S_a < S_e < S_b$.

Define the relation R over $\overrightarrow{E}_k$ as follows: given $a, b \in \overrightarrow{E}_k$, aRb iff *i)* a is incoming edge of some $p \in V$ and b is outgoing edge of p, i.e. $R_a = L_b$, *ii)* $S_a > S_b$ and *iii)* there's no other $c \in \overrightarrow{E}_k$ s.t. $S_a > S_c > S_b$. It is simple to verify by the previous proposition that aRc and bRc implies $a = c$. Define the equivalence relation R^* as the reflexive, symmetric, transitive closure of R. R^* partitions $\overrightarrow{E}_k$ into classes, each one consisting of one chain of non-overlapping directed edges going from left to right. Such chains are called *convex chains* of G_k.

Lemma 3.4 ([8]). Let $\overrightarrow{E}_k / R^* = \{C_1, ..., C_m\}$ be the set of convex chains obtained partitioning $\overrightarrow{E}_k$ by R^*. Each chain C_i has a unique leftmost endpoint (i.e. endpoint with no incoming edge), and there are $k + 1$ leftmost endpoints in G_k.

Corollary 3.5. There are at most $k + 1$ convex chains partitioning $\overrightarrow{E}_k$.

Given two edges $a, b \in \overrightarrow{E}_k$ we say that the intersection point $a \cap b$ is a *crossing* if it is not an endpoint. We need to recall the following notion.

Definition 3.7 ([2]). The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of crossings in a planar drawing of G.

Planar graphs are those without crossings, hence they are exactly those graphs G with cr(G)= 0. Remind now a lower bound on the crossing number for which we have explicit and significant constants [20].

Theorem 3.6 ([20]). The crossing number of a simple graph G having n vertices and m edges is at least $\frac{1}{33.75}\frac{m^3}{n^2} - 0.9n$.

Let C_i and C_j be two distinct convex chains of G_k, and let z be a crossing between a k-set edge a of C_i and a k-set edge b of C_j, that is $z = a \cap b$. Clearly, we can define a line segment t lying above the crossing z and connecting an endpoint of a and one of b, as illustrated in Figure 3.1. We call such a line

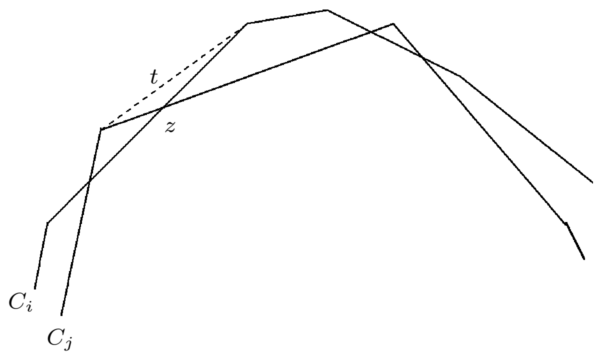

Fig. 3.1 Two k-set edges a, b, of two convex chains C_i, C_j, intersecting at a point z, that has a common tangent t to C_i, C_j, with t lying above z

segment t a *common tangent* between C_i and C_j determined by the crossing z. By exploiting Proposition 3.3 and Lemma 3.4 it is possible to demonstrate the following:

Lemma 3.7. For each common tangent t of G_k there is a unique crossing z in G_k determining t.

Lemma 3.8. There are at most nk common tangents for G_k.

Proof. Consider a vertex $p \in V$ belonging to a convex chain C_i. How many common tangents having p as left endpoint exist? Clearly, since by Corollary 3.5 there are at most k convex chains not passing through p, the answer is k at most. As the number of vertices is n, there are at most nk commont tangents. □

Corollary 3.9. There are at most nk crossings in G_k.

Proof. Simply apply Lemma 3.7. □

Before formulating the main theorem of this section we recall a useful remark by Alon and Györi [3].

Proposition 3.10. The number of $(k+1)$-sets of X is equal to the number of oriented k-set edges in E_k.

The intuitive explanation of the Proposition 3.10 is that for an oriented k-set edge $e \in E_k$, there are k points on the left of e and $n-k-2$ points on the say right of e. We can conventionally consider one endpoint, say the head, belonging to the left half-plane, and the other endpoint, say the tail, belonging to the right half-plane, so that the right half-plane conventionally has $n-k-1$ points and the left half-plane conventionally has $k+1$ points, i.e. is a $(k+1)$-set. Finally, we have the desired bound:

Theorem 3.11 ([8]). Given a set X of n points in $\mathbb{R}^2$ the number of $(k+1)$-sets of X is less than $f_{k+1}(n) = 6.48n\sqrt[3]{(k+1)}$.

Proof. Construct the k-graph $G_k = \langle V, \overrightarrow{E}_k \rangle$ as explained above and denote $n = |V| = |X|, m = |\overrightarrow{E}_k|$. By Theorem 3.6 there are at least $(1/33.75)m^3/n^2 - 0.9n$ crossings in G_k, and by Corollary 3.9 there are at most nk crossings in G_k; hence $(1/33.75)m^3/n^2 - 0.9n \leq nk$. It holds immediately that $m < 3.24n\sqrt[3]{(k+1)}$. By recalling that the number of oriented k-set edges is $|E_k| < 2|\overrightarrow{E}_k| = 2m$, the claim follows by Proposition 3.10. □

Written at the end of the last proof there is the effective inequality that we will use later on.

Corollary 3.12. The number of oriented k-set edges is $|E_k| < 6.48n\sqrt[3]{(k+1)}$.

3.4 Size Constrained 2-Clustering in the Plane

In the remaining part of this chapter we present an efficient technique that, taking in input a finite set X of points and an integer $k > 1$, finds an optimal 2-clustering $\{A, \bar{A}\}$ of X with constraint $|A| = k$, for the Euclidean norm. We will see that this algorithm works in time $O(n^2\sqrt[3]{k}\log^2 n)$.

We will make use of the abstract operations on certain convex closures, which will be specified in the next section. Before describing the technique we need some preliminaries.

Given two disjoint polygons it is easy to see that there exist 4 straight lines which are tangent to both polygons: those tangents are called *bitangents*. Two bitangents keep one polygon on one side and the other polygon on the other side, while the other two bitangents keep both polygons on the same side. Bitangents as well as straight lines can be oriented.

Given $X \subset \mathbb{R}^2$ and two points $a, b \in X$ we denote by (a, b) the oriented edge from a to b and also the oriented straight line throught a, b with little abuse of language. We define

$$X^+(a, b) = \{x \in X : x \text{ is in the open-half plane on the right of } (a, b)\}$$

that is the set of points on the right-hand side of (a, b); analogously we define $X^-(a, b)$ as the set of points on the left-hand side of (a, b).

Given $X \subset \mathbb{R}^2$ in general position, we say that the edge (a, b) with $a, b \in X$, $a \neq b$, is a $(k-1)$-*set edge* if the points $A' = X^+(a, b)$ on the right-hand side of (a, b) are exactly $k - 1$ (and hence the points $B' = X^-(a, b)$ are exactly $n - k - 1$).
Setting $A = A' \cup \{a\}$ and $\bar{A} = B' \cup \{b\}$, we can observe that:

1. A is a k-set of X;
2. the straight line (a, b) is the unique bitangent between $\mathrm{Conv}(A)$ and $\mathrm{Conv}(\bar{A})$ that keeps $A \smallsetminus \{a\}$ on its right.

Indeed as illustrated in Figure 3.2, two of the other three bitangents between

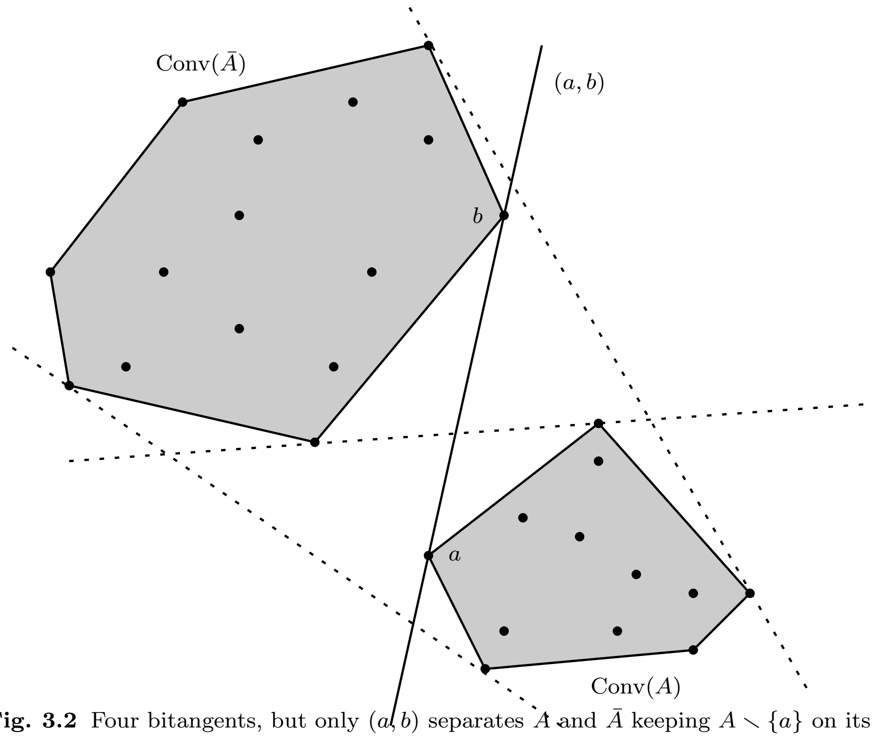

Fig. 3.2 Four bitangents, but only (a, b) separates A and $\bar{A}$ keeping $A \smallsetminus \{a\}$ on its right-hand side.

$\mathrm{Conv}(A)$ and $\mathrm{Conv}(\bar{A})$ does not separate A from $\bar{A}$, and the remaining one does not keep $A \setminus \{a\}$ on its right-hand side.

Fact 3.13. We have established a bijection between the $(k-1)$-set edges of X and the k-sets of X:

$$\begin{aligned}(a,b) &\longmapsto \{a\} \cup X^+(a,b)\\ A &\longmapsto \text{unique edge } (a,b) \text{ of } X \text{ which is bitangent to } \mathrm{Conv}(A)\\ &\qquad \text{and } \mathrm{Conv}(\bar{A}) \text{ and keeps } A \setminus \{a\} \text{ on the right}\end{aligned}$$

Denote by $A(a,b)$ the k-set corresponding to the $(k-1)$-set edge (a,b), and by $(a(A), b(A))$ the $(k-1)$-set edge corresponding to the k-set A.

Now, let $E_{k-1} = \{(a_1,b_1), ..., (a_H,b_H)\}$ be the set of $(k-1)$-set edges of X; it can be assigned the cyclic order $<$ obtained by enumerating the vectors

$$(b_1 - a_1), ..., (b_H - a_H)$$

in couterclock-wise sense. W.l.o.g. suppose $(a_1,b_1) < (a_2,b_2) < ... < (a_H,b_H)$ and define the *next bitangent* operation:

$$(a_{\langle i+1\rangle_H}, b_{\langle i+1\rangle_H}) =: \mathrm{NextBitangent}(a_i,b_i)$$

where we denoted by $\langle i\rangle_H$ the remainder of the integer division of i by H. If (a_i,b_i) is the bitangent associated to $A(a_i,b_i)$, then (a_{i+1},b_{i+1}) turns out to be the bitangent associated to $A(a_i,b_i) \cup \{b_i\} \setminus \{a_i\}$, that means

$$A(a_{i+1},b_{i+1}) = A(a_i,b_i) \cup \{b_i\} \setminus \{a_i\}.$$

Hence, setting

$$S_i = \sum_{x \in A(a_i,b_i)} x$$

the following equality holds:

$$S_{\langle i+1\rangle_H} = S_i - a_{\langle i+1\rangle_H} + b_{\langle i+1\rangle_H}$$

We hence observe that S_{i+1} can be easily computed, given S_i, with a constant number of operations. The optimal 2-clustering of X can then be obtained by the Algorithm 3.

Algorithm 3 needs some explanation. It starts by generating an initial $(k-1)$-set edge by exploiting the following

Proposition 3.14. Given the point set X with distinct y-coordinates, an oriented $(k-1)$-set edge can be obtained in $O(n)$ time.

Proof. By applying a $O(n)$ time selection algorithm [6] on X we can obtain the point p_0 with the k-th smallest y-coordinate. Then we choose the edge $e_0 = (p_0, q_0)$ with slope nearest to horizontal, i.e.

Algorithm 3 2-RCC

```
Input: a set $X \subset \mathbb{R}^2$ of $n$ points in general position; an integer $1 \leq k \leq \lfloor n/2 \rfloor$
Output: the solution $\pi = \{A, B\}$ of the 2-RCC on $X$ with constraint $\mathcal{M} = \{k, n-k\}$
        and Euclidean norm
1  $a_0$ = Selection($X$,$k$);
2  $b_0 = \arg\min_{b \in X : b \neq a_0}$(slope coefficient of straight line through $a_0, b$)
3  $(x, y) = (a_0, b_0)$
4  $S = \sum_{x \in A(a_0, b_0)} x$
5  $q = \|S\|^2$
6  $(a, b)$ = NextBitangent($a_0, b_0$)
7  while $(a, b) \neq (a_0, b_0)$ do
8      $S = S - a + b$
9      if $q < \|S\|^2$ then
10         $q = \|S\|^2$
11         $(x, y) = (a, b)$
12     $(a, b)$ = NextBitangent($a, b$)
13 $\pi = \{X^+(x, y) \cup \{x\}, X^-(x, y) \cup \{y\}\}$
14 return $\pi$
```

$$q_0 = \arg\min_{q \in X : q \neq p_0} (\text{slope coefficient of the straight line through } p_0, q)$$

It is easy to see that e_0 determines a k-set, hence can be arbitrarily oriented. The point q_0 can be obtained in $O(n)$ time. □

The NextBitangent operation called at lines 6 and 12 is the abstract operation, introduced above, for obtaining the successive $(k-1)$-set edge in the cyclic order of E_{k-1}. An effective procedure for doing this operation will be explained in the next section.
The update operations inside the while-loop are simplified by the formulation (3.5).

It is important to observe that, supposing to implement the NextBitangent operation in 1 step, the computational time of the Algorithm 3 is

$$O(H) = O(n\sqrt[3]{k})$$

since the number H of $(k-1)$-set edges is bounded by $6.48n\sqrt[3]{k}$ due to Corollary 3.12.

To complete the time complexity analysis it remains to describe in the next section the effective procedure for the NextBitangent operation, that will exploit an efficient dynamic data structure for maintaining the convex hull of a point set in the plane.

3.5 Dynamic Convex Hull

In this section we illustrate the problem of designing an efficient data structure for constructing and maintaining the convex hull of a planar point set under the operations of insertion, deletion and query.
Moreover, we will see how this efficient data structure can be applied to give a bound on the complexity of the algorithm described in Section 3.4. In particular we can give a bound on the number of operations performed by NextBitangent among all calls to it.

We recall some basic notions. A set $A \subseteq \mathbb{R}^d$ is *convex* iff for any $x, y \in A$ it holds $\lambda x + (1-\lambda)y \in A$ for all $0 \leq \lambda \leq 1$. The intersection of a collection of convex subsets of the Euclidean space $\mathbb{R}^d$ is itself convex [24].

Definition 3.8. Given a set A in $\mathbb{R}^d$ the *convex closure* or *convex hull* of A, denoted by $\mathrm{Conv}(A)$, is the smallest convex subset of $\mathbb{R}^d$ containing A:

$$\mathrm{Conv}(A) = \bigcap \{X \subseteq \mathbb{R}^d : A \subseteq X, X \text{ is convex}\}$$

Clearly, in $\mathbb{R}^1$ the convex closure of A is the closed interval $[\inf A, \sup A]$. It is well-known that the convex closure of a finite set A of points in $\mathbb{R}^d$ is a polytope [22] described by an intersection of finitely many half-spaces; in particular in $\mathbb{R}^2$, $\mathrm{Conv}(A)$ is a convex polygon.
In $\mathbb{R}^2$ it is possible to identify a polygon by simply giving its vertices, hence the determination of the convex closure $\mathrm{Conv}(A)$ of a given set $A \subset \mathbb{R}^2$ reduces to finding the vertices of the associated polygon.

Manipulating the convex closure of a set is a very common task when dealing with geometric techniques. Although the definition is not very constructive, there are very efficient algorithms for obtaining the convex closure of a given finite set of points both for $d = 2$ and $d = 3$. In this section we will recall some techniques in the planar case.

The convex hull problem we study in this section is formalised as follows.

Definition 3.9 (Convex Hull Problem (CH)). Given a finite set $X = \{x_1, ..., x_n\}$ of n points in $\mathbb{R}^2$, find the sequence of vertices $v_1, ..., v_h$ of the convex closure $\mathrm{Conv}(X)$ in clock-wise order.

Classical algorithms to solve the CH problem are the Graham's Scan and the Jarvis' March [6]; the first one works in $O(n \log n)$ time and $O(n)$ space, while the second one works in time $O(hn)$, where h is the number of vertices of the convex closure, and space $O(n)$. Moreover, there exists an algorithm for constructing the convex closure of a set by successive insertions of its elements, each taking $O(\log n)$ time [21], and hence working in time $O(n \log n)$; this algorithm has evident applications in real-time problems.

The algorithms for CH strictly depend on the sorting of some objects; this implies that the CH is as difficult as the sorting problem, indeed it is well-known that the sorting problem is linear-time reducible to CH, and this entails that CH is solvable in time $\Omega(n \log n)$.

A more challanging question is the design of an efficient data structure for maintaining the convex closure of a set A of points accomodating the operations:

- insert(A,x): given x determine the convex closure of $A \cup \{x\}$,
- delete(A,x): given x determine the convex closure of $A \setminus \{x\}$,
- query(A,x): given x decide whether $x \in \mathrm{Conv}(A)$.

A first efficient solution for this task was given by Overmars and Van Leeuwen [17]. Later, an algorithm working in $O(\log n)$ amortized time per each deletion and insertion were proposed [4]. It remains an open problem whether it is possible to design a data structure allowing insertion and deletion operations in (non-amortized) time $O(\log n)$. In this section we make use of the first solution.

Let $X = \{x_1, ..., x_n\} \subset \mathbb{R}^2$. The algorithm of Overmars and Van Leeuwen keeps in memory two partial hulls: the convex closure of $X \cup \{(+\infty, 0\}$ (i.e. adding to X the improper point $x = +\infty$) called left hull, and the convex closure of $X \cup \{(-\infty, 0\}$ (X with the improper point $x = -\infty$) called right hull. The main feature of the algorithm is the ability to merge the left (resp. right) hull of two sets A and B to obtain the left (resp. right) hull of $A \cup B$ in time $O(\log n)$.
To maintain the left and right hull of X the authors suggested the use of balanced search trees, such as Red-Black, AVL or Weight Balanced Trees [23].

Furthermore, each node of such tree stores into an additional data structure the inner points of a left (resp. right) hull, i.e. points that do not belong to the boundary. Such a data structure must adhere to the interface of operations called "concatenable queue" [1].

A concatenable queue is a data structure that stores an ordered dictionary Q that allows, besides the usual operations of insertion, deletion, minimum and element search, also the operations:

- concatenate(Q_1, Q_2): takes in input two concatenable queues Q_1 and Q_2 such that $q_1 \leq q_2$ for $q_1 \in Q_1, q_2 \in Q_2$ and gives the concatenation of Q_1 and Q_2 respecting the order,
- split(Q, x): factorise the concatenable queue Q into two concatenable queues Q_1 and Q_2 such that $q_1 \leq x < q_2$ for $q_1 \in Q_1, q_2 \in Q_2$.

A possible implementation of such an interface is by means of the 2-3 Trees, which leads to a $O(\log n)$ time complexity for a single operation of the concatenable queue.

Since for each operation of insertion and deletion into a left (resp. right) hull an operation of the concatenable queue must be performed, it follows:

Theorem 3.15 ([17]). There exists an algorithm to maintain the convex hull of a set of n points in the plane at a time cost of $O(\log^2 n)$ per insertion and deletion.

Therefore, n operations on the convex hull of a set of n points cost $O(n \log^2 n)$. Moreover, we remind that for such maintained convex hull we can obtain in $O(1)$ time the outgoing edge of a point on the boundary, or equivalently the successor $\mathrm{Succ}(v)$ of a vertex v.

An efficient implementation of the algorithm of Overmars and Van Leeuwen, using modified Red-Black Trees for the balanced search trees and 2-3 Trees for the concatenable queue, was written in C programming language in [27].

Our technique for finding the bitangent of two polygons is based on the maintenance of the convex hull corresponding to the respective polygons. Recall from Section 3.4 that the problem consists in

Definition 3.10 (Next Bitangent Problem). Given two convex hulls $\mathrm{Conv}(A_i)$ and $\mathrm{Conv}(\bar{A}_i)$ and their associated bitangent (a_i, b_i), find the bitangent (a_{i+1}, b_{i+1}) associated to $A_i \smallsetminus \{a_i\} \cup \{b_i\}$ and $\bar{A}_i \smallsetminus \{b_i\} \cup \{a_i\}$.

As in Section 3.4 we take the set A_i as a k-set, hence also $A_i \smallsetminus \{a_i\} \cup \{b_i\}$ is, and both (a_i, b_i) and (a_{i+1}, b_{i+1}) are $(k-1)$-set edges. Hence the problem turns out to be equivalent to finding the successive $(k-1)$-set edge in the cyclic order already introduced on E_{k-1}.

Such kind of problem was already studied by Edelsbrunner and Welzl [11] in the dual plane by means of a "sweep line" method on the k-belt, which basically consists in a region of the dual plane bounded by a k-level and a $(n-k)$-level (see Section 3.3). They gave an efficient solution to the Next Bitangent Problem formulated in the dual plane and, as a consequence, Proposition 3.16 holds.

Here, we propose an alternative technique to solve this problem directly in the primal plane by means of proper paths formed by k-set edges; this technique attains the same time complexity of the sweep line method in [11]. We now explain the details.

Recall the notation of Section 3.4, and denote by $A_i = A(a_i, b_i)$ the k-set associated to (a_i, b_i), and by $\bar{A}_i = \overline{A(a_i, b_i)}$ its complement; denote by $\mathrm{Succ}(a)$ the direct successor of a vertex a on a convex hull, i.e. the vertex following a in counterclock-wise order on the boundary of the convex hull. Suppose that we maintain the convex closures $\mathcal{A}_i = \mathrm{Conv}(A_i)$, $\bar{\mathcal{A}}_i = \mathrm{Conv}(\bar{A}_i)$; the convex closures $\mathcal{A}_{i+1}$ and $\bar{\mathcal{A}}_{i+1}$ of $A_{i+1} = A_i \smallsetminus \{a_i\} \cup \{b_i\}$ and $\bar{A}_{i+1} = \bar{A}_i \smallsetminus \{b_i\} \cup \{a_i\}$ can be computed in time $O(\log^2 n)$ by Theorem 3.15. Notice that (a_i, b_i) is also a bitangent to $\mathcal{A}_{i+1}$ and $\bar{\mathcal{A}}_{i+1}$. To obtain the bitangent (a_{i+1}, b_{i+1}) associated to A_{i+1} we proceed as follows.

First, notice that there exist 2 consecutive edges $(a, a'), (a', a'')$ on the convex closure $\mathcal{A}_{i+1}$, such that:

- the straight lines (a, a') and (a', a'') cannot cross the tangency point b_{i+1} on the convex closure $\bar{\mathcal{A}}_{i+1}$, by the general position hypothesis;
- the intersection of the right-hand side of (a, a') with the left-hand side of (a', a'') contains the bitangency point b_{i+1}; it may be that $a = b_i$.

Hence, we can surely state that a' is the other tangency point a_{i+1} on the convex closure $\mathcal{A}_{i+1}$.

Second, let $a' = a_{i+1}$ be the tangency point on $\mathcal{A}_{i+1}$; there exist 2 consecutive edges $(b, b'), (b', b'')$ on $\bar{\mathcal{A}}_{i+1}$ such that their associated straight lines do not cross a', and it holds that the sequence $(b', b''), (b', a'), (b, b')$ is in order w.r.t. slope (defined in Section 3.2) if and only if b' is the tangency point b_{i+1}.

Therefore, these two considerations suggest an idea for an algorithm, that intuitively works as follows:

1. start with the edges $(a = a_i, a')$ on the boundary of $\mathrm{Conv}(A_{i+1})$ and $(b = b_i, b')$ on the boundary of $\mathrm{Conv}(\bar{A}_{i+1})$;
2. jump to the direct successor (a', a'') on $\mathrm{Conv}(A_{i+1})$ and from (b, b') go through its consecutive successors $(\beta, \beta'), (\beta', \beta'')$ checking whether (β', β''), (b', a'), (β, β') are in order w.r.t. slope;
3. if this is the case then (a', b') is the bitangent of $\mathrm{Conv}(A_{i+1})$ and $\mathrm{Conv}(\bar{A}_{i+1})$, otherwise continue going through the successors of (b, b') if the slope of the successor is greater than the slope of (a', a'').

Then iterate again on these steps on the successor of (a', a'') and so on.

We formalise this idea in the Algorithm 4. Preliminarly, we notice that the algorithm must store, as permanent structures, the convex hulls $\mathcal{A}_i$ and $\bar{\mathcal{A}}_i$ so that they are available each time NextBitangent is called. Moreover, it is easy to see that the edges (a, a') considered in the iterations of the algorithm are $(k-2)$-set edges; moreover, these edges are considered in order w.r.t. the slope. Analogously, the edges (b, b') considered in the iterations are $(n-k-2)$-set edges and are considered in order w.r.t. the slope. Recalling the bound on the k-set edges (Corollary 3.12) and Remark 3.2, we can then conclude the following

Proposition 3.16. $\lfloor 6.48 n \sqrt[3]{k} \rfloor$ calls to the NextBitangent algorithm for two convex hulls of size k and $n-k$ requires $(f_{k-2}(n) + f_{k-n-2}(n)) \cdot O(\log^2 n) = O(n\sqrt[3]{k} \cdot \log^2 n)$ time.

As a consequence, since the remainder of the while-loop in the Algorithm 3 requires $O(1)$ time per each iteration, the total time complexity is dominated by the executions of the NextBitangent procedure.

Theorem 3.17. The Algorithm 3 for the 2-clustering with size constraint k in the plane with Euclidean norm requires $O(n\sqrt[3]{k}\log^2 n)$ time.

Algorithm 4 NextBitangent

```
Input: a $(k-1)$-set edge $(a_i, b_i)$
Output: the successive $(k-1)$-set edge $(a_{i+1}, b_{i+1})$ in the cyclic order of $E_{k-1}$
$\mathcal{A}_{i+1} = \mathrm{Conv}(\mathcal{A}_i \setminus \{a_i\} \cup \{b_i\})$        // one insertion and one deletion
$\bar{\mathcal{A}}_{i+1} = \mathrm{Conv}(\bar{\mathcal{A}}_i \setminus \{b_i\} \cup \{a_i\})$
$a = b_i$; $b = a_i$
$a' = \mathrm{Succ}(a)$; $b' = \mathrm{Succ}(b)$
found_a = false
repeat
    while slope of $(a', a)$ < slope of $(b, b')$ and found_a = false do
        $a = a'$
        $a' = \mathrm{Succ}(a')$
        if slope of $(a, a')$ > slope of $(b, a)$ then
            $a_{i+1} = a$
            found_a = true
    $b = b'$
    $b' = \mathrm{Succ}(b')$
until slope of $(b, b')$ > slope of $(a, b)$
$b_{i+1} = b$
return $(a_{i+1}, b_{i+1})$
```

References

1. A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley Pub. Co., 1974.
2. M. Ajtai, V. Chvátal, M. M. Newborn, and E. Szemerédi. Crossing-free subgraphs. *Annals of Discrete Mathematics*, 12:9–12, 1982.
3. N. Alon and E. Györi. The number of small semispaces of a finite set of points in the plane. *J. Comb. Theory Ser. A*, 41:154–157, January 1986.
4. G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd IEEE Sympos. Found. Comput. Sci*, pages 617–626, 2002.
5. B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25:76–90, June 1985.
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.
7. T. Dey. Improved bounds on planar k-sets and k-levels. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 156–161, October 1997.
8. T. Dey. Improved Bounds for Planar k-Sets and Related Problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.
9. H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* EATCS monographs on theoretical computer science. Springer, 1987.
10. H. Edelsbrunner and E. Welzl. On the Number of Line Separations of a Finite Set in the Plane. *Journal of Combinatorial Theory, Series A*, 38(1):15–29, 1985.
11. H. Edelsbrunner and E. Welzl. Constructin Belts in Two-Dimensional Arrangements with Applications. *SIAM J. Comput.*, 15(1):271–284, February 1986.
12. P. Erdős, L. Lovász, A. Simmons, and E. G. Straus. Dissection graphs of planar point sets. In *A survey of combinatorial theory (Proc. Internat. Sympos., Colorado State Univ., Fort Collins, Colo., 1971)*, pages 139–149. North-Holland, Amsterdam, 1973.
13. P. Hansen, B. Jaumard, and N. Mladenovic. Minimum Sum of Squares Clustering in a Low Dimensional Space. *Journal of Classification*, 15:37–55, 1998.

14. M. Klawe, M. Paterson, and N. Pippenger. Inversions with $n2^{1+\Omega(1+\sqrt{\log n})}$ transpositions at the median. Unpublished manuscript, 1982.
15. F. T. Leighton. *Complexity Issues in VLSI.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1983.
16. L. Lovász. On the number of halving lines. *Ann. Univ. Sci. Budapest, Eötvös, Sec. Math*, 14:107–108, 1971.
17. M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23(2):166 – 204, 1981.
18. J. Pach, W. Steiger, and E. Szemerédi. An upper bound on the number of planar k-sets. In *30th Annual Symposium on Foundations of Computer Science (FOCS 1989)*, pages 72–79. IEEE Computer Society, 1989.
19. J. Pach, W. Steiger, and E. Szemerédi. An Upper Bound on the Number of Planar K-Sets. *Discrete & Computational Geometry*, 7:109–123, 1992.
20. J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439, 1997. 10.1007/BF01215922.
21. F. Preparata. An Optimal Real-Time Algorithm for Planar Convex Huls. *Communications of the ACM*, 22(7):402–405, July 1979.
22. F. Preparata and M. Shamos. *Computational geometry: an introduction.* Texts and monographs in computer science. Springer-Verlag, 1985.
23. E. Reingold, J. Nievergelt, and N. Deo. *Combinatorial algorithms: theory and practice.* Prentice-Hall, 1977.
24. R. T. Rockafellar. *Convex Analysis.* Princeton University Press, 2nd printing edition, 1972.
25. M. Sharir, S. Smorodinsky, and G. Tardos. An Improved Bound for k-Sets in Three Dimension. *Discrete & Computational Geometry*, 26(2):195–204, 2001.
26. G. Tóth. Point Sets with Many k-Sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001.
27. L. Zanetti. Struttura Dati Dinamica per Convex Hull nel Piano. Thesis, University of Milan, Milan, 2011.

Chapter 4
Clustering with norm L^p

In this chapter we want to study the 2-SCC-d problem with norm L_p, p positive integer. We firstly shall restrict our attention to even p's since the analysis is far simpler, then we extend the developed method for the case of odd p. We will prove that, by appropriately decomposing the space of the parameter of the hypersurfaces separating the 2 clusters, we obtain a set of 2-clusterings containing the optimal clustering of size m, for every constraint size m, thus allowing us to compute the optimal costs and the cardinality of the clusters, for every cluster size constraint m. This method will be proved to have polynomial time complexity with respect to the data set size n and the integer p, for a fixed dimension d.

This chapter is organised as follows: Section 4.1 recalls the separation property and illustrate the problem in the parameter space; Section 4.2 gives a richer description of the parameter space in the language of the real algebraic geometry; Section 4.3 outlines the central tool of our clustering method which allow the decomposition of the parameter space; such a tool is constructed using basic algebraic notions such as resultants, Sturm sequences and some bounds on the real roots that will be explained throughout the Sections 4.4, 4.5, 4.6; in Section 4.7 we tackle the problem of comparing the cost of two clusterings; a numerical method we develop for solving this problem will be used in Section 4.8, which is devoted to the design of the clustering algorithm for 2-SCC-d and its complexity analysis.

4.1 Separating hypersurfaces

From Chapter 3 we know that the 2-SCC problem is solvable in polynomial time for fixed dimension $d = 2$ and norm $\| \|_p$ with $p = 2$. In this chapter we will assume that p is an integer $p > 2$.

Given $X = \{x_1, ..., x_n\}$ and constraints $\{m, n - m\}$, let $\{A, B\}$ be the optimal 2-clustering with $|A| = m, |B| = n - m$. Consider the function

$$f(x;\mu,\lambda,\gamma) = ||x-\mu||_p^p - ||x-\lambda||_p^p - \gamma = 0 \tag{4.1}$$

From the Separation Property (Theorem 1.4) we know that there exist C_A, $C_B \in \mathbb{R}^d$, $c \in \mathbb{R}$ such that the hypersurface of equation

$$f(x;C_A,C_B,c) = 0$$

is well defined and separates the two clusters, i.e. the following property is satisfied.
Property $(*)$:

$$f(x_i;C_A,C_B,c) < 0 \quad \text{for all } x_i \in A, \quad f(x_j;C_A,C_B,c) > 0 \quad \text{for all } x_j \in B$$

or

$$f(x_i;C_A,C_B,c) > 0 \quad \text{for all } x_i \in A, \quad f(x_j;C_A,C_B,c) < 0 \quad \text{for all } x_j \in B$$

When the previous condition holds we also say that the parameter $C = (C_A, C_B, c)$ *separates A and B.*

This result leads to a different paradigm for solving the 2-SCC. The problem is decomposed into two subproblems:

(1) Determine the clusterings which are separable by hypersurfaces of the parametric family:

$$f(x;\mu,\lambda,\gamma) = ||x-\mu||_p^p - ||x-\lambda||_p^p - \gamma = 0$$

with parameter vector $\alpha = (\mu,\lambda,\gamma)$;

(2) Choose from such clusterings the optimal one satisfying the constraints $\{m, n-m\}$.

We emphasize that the above-mentioned hypersurfaces lie in the parameter space $\mathbb{R}^{2d+1}$ while the points of the data set are in a different space, namely $\mathbb{R}^d$.
Under the following conditions:

(i) the number of clusterings which are separable by such hypersurfaces is far less than all the 2^n admissible clusterings,
(ii) such clusterings are obtainable in an efficient manner, and
(iii) there exists an efficient algorithm that, having in input 2 clusterings, can decide which is better,

we can hope that there's a method to solve the 2-SCC problem in feasible time.

In order to determine some tools for solving the point (1), we start by considering the case of even integer $p > 2$. In this case the functions $f(x,\mu,\lambda,\gamma)$ are representable by multivariate polynomials of degree p. Let's denote the vector $\alpha = (\mu,\lambda,\gamma)$. Given $X = \{x_1, ..., x_n\}$, the algebraic varieties

$$Z_i = \{\alpha \in \mathbb{R}^{2d+1} : f(x_i;\alpha) = 0\} \qquad , i = 1, ..., n$$

are generally sets with the cardinality of continuum. However, they decompose the parameter space $\mathbb{R}^{2d+1}$ into a finite number of connected subsets $\{R_1, ..., R_s\}$ having the following so-called sign-invariance property:

$$\alpha, \alpha' \in R_j \Longrightarrow \mathrm{sgn}(f(x_i, \alpha)) = \mathrm{sgn}(f(x_i, \alpha')) \text{ for all } i = 1, ..., n.$$

In fact the varieties $Z_1, ..., Z_n$ form the boundaries of the sets $R_1, ..., R_s$. We are interested in sets R_j such that $\alpha \in R_j$ implies

$$\mathrm{sgn}(f(x_i, \alpha)) \in \{-1, +1\} \qquad \text{for all } i$$

Indeed, in this case the property $(*)$ is satisfied. Moreover, by the continuity of the functions $f(x_i, \alpha)$, for every $\alpha \in R_j$ there is an open sphere $I(\alpha)$ s.t. $\alpha \in I(\alpha) \subset R_j$: R_j contains a vector $\hat{\alpha}$ with rational dyadic components, i.e. components represented as rational numbers with denominator that is a power of 2.
Since such R_j's are semi-algebraic sets, in the next section we introduce some preliminary notions of real algebraic geometry.

4.2 Semi-algebraic sets

In this section we recall basic notions of real algebraic geometry and give some useful properties of semi-algebraic sets [3].

Let $\mathbb{R}$ be the field of real numbers and $\mathbb{R}[x]$ the ring of polynomials over $\mathbb{R}$ in d variables $x = (x_1, ...x_d)$. Given a polynomial $f \in \mathbb{R}[x]$ the set $\mathcal{Z}(f) = \{x \in \mathbb{R}^d : f(x) = 0\}$ is called the *zero set* of f. A subset E of the affine space $\mathbb{R}^d$ is called *algebraic* if it is the zero set of some $f \in \mathbb{R}[x]$. The algebraic sets are closed under finite union and finite intersection. Indeed, it holds $\mathcal{Z}(f) \cup \mathcal{Z}(g) = \mathcal{Z}(fg)$, and

$$\bigcap_{i=1}^{s} \mathcal{Z}(f_i) = \mathcal{Z}(\sum_{i=1}^{s} f_i^2)$$

since we work on the real field, which is not algebraically closed.

However, the algebraic sets are not closed under complementation and projection. On the contrary, a more stable class is that one of semi-algebraic sets. The *semi-algebraic* sets can be defined recursively as follows.

Given a polynomial $f \in \mathbb{R}[x]$ we denote by $\mathcal{Z}_-(f) = \{x \in \mathbb{R}^d : f(x) < 0\}$ the sublevel set of f. The maps $\mathcal{Z}$ and $\mathcal{Z}_-$ can be extended to collection of polynomials: $\mathcal{Z}(\mathcal{F}) = \{x \in \mathbb{R}^d : f(x) = 0 \text{ for all } f \in \mathcal{F}\}$ and $\mathcal{Z}_-(\mathcal{F}) = \{x \in \mathbb{R}^d : f(x) < 0 \text{ for all } f \in \mathcal{F}\}$.

Definition 4.1. For any $f \in \mathbb{R}[x]$, $\mathcal{Z}(f)$ and $\mathcal{Z}_-(f)$ are semi-algebraic sets. If S and S' are semi-algebraic sets, then also $S \cup S'$ and $S \cap S'$ are semi-algebraic sets.

Substantially, a semi-algebraic set is the set of solutions of a boolean combination of equalities and inequalities, i.e. a system of the form

$$\bigvee_{i=1}^{r} \bigwedge_{j=1}^{s_i} (f_{i,j} \bowtie_{i,j} 0)$$

where for each i, j the polynomial $f_{i,j} \in \mathbb{R}[x]$ and the relation $\bowtie_{i,j} \in \{<, =\}$. Obviously, algebraic sets are also semi-algebraic, but not conversely. The algebraic sets form the closed sets of a special topology, called the *Zariski topology* [15].

The class of semi-algebraic sets is closed under complementation as it can be easily seen by

$$\overline{\mathcal{Z}(f)} = \mathcal{Z}_-(f) \cup \mathcal{Z}_-(-f)$$

and is closed under projection as stated by the following:

Proposition 4.1. Let $\pi : \mathbb{R}^{n+1} \to \mathbb{R}^n : (x_1, ..., x_{n+1}) \mapsto (x_1, ..., x_n)$ be the projection on the first n coordinates. If $A \subseteq \mathbb{R}^{n+1}$ is semi-algebraic then also the image of A under π, i.e. $\pi(A) = \{(x_1, ..., x_n) : \exists x_{n+1}(x_1, ..., x_{n+1}) \in A\}$, is semi-algebraic.

Remark 4.1. It can be easily proven that a semi-algebraic set is the projection of some algebraic set. Indeed, the solutions of the inequality $f(x) < 0$ are the projection of the solutions of the equation $y^2 f(x) = 1$.

Remark 4.2. Whilst the Proposition 4.1 is an essential property of semi-algebraic sets, this doesn't hold true for the class of algebraic sets. For example, the algebraic curve A defined by the zero set of the polynomial $f(x, y) = y^2 - x$:

$$A = \mathcal{Z}(f) = \{(x, y) \in \mathbb{R}^2 : y^2 - x = 0\}$$

has the image $\pi(A)$ under the projection $\pi : (x, y) \mapsto x$ described by:

$$\pi(A) = \{x \in \mathbb{R} : x \geq 0\}$$

which is semi-algebraic but not algebraic, since it cannot be expressed as zero set of any polynomial in x.

The theory of real algebraic geometry dealing with the semi-algebraic sets was initially introduced by algebraist, and then developed by logicians to tackle several important problems in the theory of real closed fields, such as the general decision problem and the quantifier elimination problem [2]. We briefly recall such an approach.

A *first-order formula* $\Phi(x)$ of the language of ordered fields with coefficients in $\mathbb{R}$ is a first-order formula with free variables $x = (x_1, ..., x_d)$, quantified variables $y = (y_1, ..., y_e)$ and atoms of the kind $P(x, y) = 0$ or $P(x, y) < 0$, where P is a polynomial over $\mathbb{R}$. Unless otherwise stated, we will always mean this kind of first-order formula. The $\mathbb{R}$-*realisation* (or simply *realisation*) of a first-order formula $\Phi(x)$ with free variables x is the set

$$\mathcal{R}_{\mathbb{R}}(\Phi) = \{x \in \mathbb{R}^d : \Phi(x) \text{ is true}\}$$

It is well known that $\mathcal{R}_{\mathbb{R}}(\Phi)$ is a semi-algebraic set [2]. The *quantifier elimination problem* for a given quantified formula

$$\Phi(x) = (Q_1 y_1)...(Q_e y_e)F(x_1, ..., x_d, y_1, ..., y_e)$$

consists in finding a quantifier-free formula $\Psi(x)$, such that Φ and Ψ have the same $\mathbb{R}$-realisation. The *general decision problem* consists in deciding the truth value of a given *Tarski sentence* Φ, i.e. a first-order formula without free variables,

$$\Phi = (Q_1 y_1)...(Q_e y_e)F(y_1, ..., y_e)$$

where each Q_i is either $\exists$ or $\forall$. It is clear that the general decision problem is a particular case of the quantifier elimination problem, in fact when $d = 0$. A special case of the general decision problem, in turn, is the *decision problem for the existential theory of the reals*, which consists in deciding the truth value of a given Tarski sentence with only existential quantifiers, namely $Q_i = \exists$ for all $i = 1, ..., e$. Thus, it is easy to see that the latter problem is equivalent to deciding whether a given semi-algebraic set is empty or not.

The main tool developed in the early 50's to face this kind of problem is the Tarski-Seidenberg principle that we formulate in a simplied form. Here we denote by σ a vector of signs, i.e. $\sigma = (\sigma_1, ..., \sigma_s) \in \{-1, 0, +1\}^s$, and by x a vector of $d + 1$ variables $(x_0, x_1, ..., x_d)$.

Theorem 4.2 (Tarski-Seidenberg Principle). Let $f_1, ..., f_s \in \mathbb{R}[x]$ be a sequence of polynomials in $d + 1$ variables. Given a vector $\sigma = (\sigma_1, ..., \sigma_s)$ of s signs, there exists a quantifier-free formula $\Phi_\sigma(x_1, ..., x_d)$ such that, for every $a_1, ..., a_d \in \mathbb{R}$ it holds: $\Phi_\sigma(a_1, ..., a_d)$ is true if and only if $\exists a_0 \in \mathbb{R}$ s.t. $(\bigwedge_{i=1,...,s} \operatorname{sgn} f_i(a_0, a_1, ..., a_d) = \sigma_i)$, that is the system

$$\begin{cases} \operatorname{sgn}(f_1(x_0, a_1, ..., a_d)) = \sigma_1 \\ \qquad\qquad \vdots \\ \operatorname{sgn}(f_s(x_0, a_1, ..., a_d)) = \sigma_s \end{cases}$$

admits a solution a_0 in $\mathbb{R}$.

It can be shown that the Proposition 4.1 relies directly on the Tarski-Seidenberg Principle.

Once semi-algebraic sets are introduced it is natural to define certain analytic or topological properties on such sets or functions defined over them, which constitute typical object of study in the context of real algebraic geometry [3].

Definition 4.2. Given two semi-algebraic sets A and B, a function $f : A \to B$ is said to be *semi-algebraic* iff its graph $\text{Graph}(f) = \{(x, f(x)) : x \in A\}$ is semi-algebraic.

It is well known that the class of semi-algebraic functions is closed under composition. Moreover, image and inverse image of semi-algebraic sets under semi-algebraic functions are semi-algebraic.

Definition 4.3. A semi-algebraic set A is called *semi-algebraically connected* iff it cannot be written as disjoint union $A = A_1 \sqcup A_2$ of two closed sets A_1, A_2 in A.

If A can be written in the above manner and A_1, A_2 are semi-algebraic and semi-algebraically connected, then they are called *semi-algebraically connected components* of A. More generally, the following result holds:

Theorem 4.3 ([3]). Every semi-algebraic set A can be written as disjoint union

$$A = \bigsqcup_{1}^{m} A_i$$

of a finite number m of semi-algebraic semi-algebraically connected sets A_i, which are clopen (i.e. closed and open) in A and are termed *semi-algebraically connected components* of A.

Remark 4.3. We introduced the notion of real algebraic sets based on polynomials in $\mathbb{R}[x]$; analogous notion can be introduced by considering polynomials in $\mathbb{C}[x]$, where $\mathbb{C}$ is the set of complex numbers, thus giving the *complex algebraic sets.*

Many concepts of complex algebraic geometry can be analysed by means of computer algebra tools. Unfortunately, these methods cannot be easily extended to cover the real case. The remainder of this section is aimed at illustrating a critical example of this assertion.

Definition 4.4. A complex algebraic set A is said to be *irreducible* iff it cannot be written as union of two proper (complex) algebraic subsets, that is when $A = A_1 \cup A_2$ with A_1, A_2 algebraic then $A_1 = A$ or $A_2 = A$. If A is not irreducible, it is said *reducible.*

Some authors [15] use the term algebraic variety to name irreducible algebraic sets, others use the term algebraic variety to refer to algebraic sets and add the word irreducible when needed. We prefer the former convention.

Theorem 4.4 ([21]). A complex algebraic variety is connected.

For a complex algebraic set the notion of irreducibility corresponds to the fact that the associated ideal I is *prime*, i.e. I is a proper ideal satisfying: $ab \in I \Rightarrow a \in I$ or $b \in I$.

Theorem 4.5 ([11]). Let V be a complex algebraic set. V is irreducible if and only if the ideal $I(V) = \{f \in \mathbb{C}[x_1, ..., x_d] : f(V) = \{0\}\}$ of polynomials vanishing on V is prime.

Hence, in the case of underlying complex field, by Theorems (4.4) and (4.5) it is sufficient to find the prime components of the associated ideal $I(A)$ in order to find the connected components of A [11]. We will now see that the real case doesn't show this property.

Example 4.1. The real cubic $x + y^2 - x^3 = 0$ shown in Figure 4.1, called

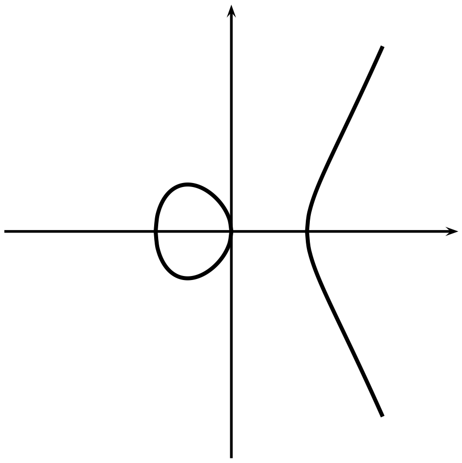

Fig. 4.1 The Weierstrass elliptic curve: $y^2 = x^3 - x$, is irreducible but has two connected components

Weierstrass equation [22], is an elliptic curve with two connected components and is irreducible.

This example confirms that the algebraic closure of the ambient field is a necessary condition for applying the last considerations, that would have greatly simplified the treatment of the problem in the space of parameters α we introduced in Section 4.1.

4.3 Cylindrical Decomposition

In this section we illustrate an effective method for the decomposition introduced in Section 4.1, which allows us to investigate the space of parameters

α of the separating hypersurface family. This method is also suitable for solving the general decision problem and the quantifier elimination problem we presented in Section 4.2.

In 1948 Tarski [23] presented a quantifier elimination method for the theory of elimination in real closed fields. Seidenberg [20], Cohen [6] and Coste [10] have successively published other methods that were doubly exponential in the number d of variables (dimension), but polynomial both in the number m of polynomials and in the maximum degree g of the polynomials, for a fixed d. Since 1973 Collins [8] has discovered and presented a new method which is polynomial time in m, g and τ, where τ is the bitsize of the coefficients, for a fixed d, but remains double exponential in d. Later, several algorithms running in single exponential time in d were introduced [5, 14], as well as other improvements for computing the solution by exploiting favourable conditions of smoothness [16]. Algorithms for solving the quantifier elimination problem have several applications such as robot motion planning [5, 4], testing termination in term-rewriting systems [1], surface construction in solid modelling [13], identification of polynomial parametric models in statistics [12]. Here we will present a novel application to the constrained clustering problem.

In this work we outline a simplified version of the method of Collins [8] mainly known as *Cylindrical Algebraic Decomposition* (CAD), that intuitively decomposes the affine space into a finite number of disjoint connected semi-algebraic regions of various dimension, each one having the property that the polynomials evaluated in all points of the region have constant sign.

Basically, the CAD consists of two phases, called *projection* and *lifting*. In the projection phase the common roots and the critical points of the given set of polynomials are repeatedly projected onto a subspace by eliminating a variable, ending with the last projection onto the real line; the projections are done orthogonally, hence the term *cylindrical*. The lifting phase takes repeatedly a decomposition of a subspace and gives a decomposition of the space above by searching for all points of the variety above and below each component of the subspace. It turns out that these points subdivides the cylinder into “slices”. The slices of the last lifting constitutes the CAD of the space.

For our aims, it is sufficient to give a simplified version of Collins’ CAD, which we abbreviate with SCAD, and differs in particular by the following characteristics which will be illustrated later on in this section:

1) We do not impose that the polynomial collections F_i is not closed under the derivation operation, and thus the elimination of the variables is done by means of the resultant instead of the subresultant [8].
2) We consider only “slices” admitting representative points with rational components, that hence can be lifted easily.

Let's first introduce some basic notions of the cylindrical algebraic decomposition, that can be found mainly in [1]. Let $\mathbb{R}^d$ be the d-dimensional real affine space; we call a non-empty connected subset $R \subset \mathbb{R}^d$ a *region* and we say that the subset

$$\mathrm{Cyl}(R) = R \times \mathbb{R}$$

of $\mathbb{R}^{d+1}$ is the *cylinder over* R.

Given a continuous real function $f : \mathbb{R}^d \to \mathbb{R}$, an f-*section* (or simply *section*) of $\mathrm{Cyl}(R)$ is the region $\mathrm{Graph}_R(f) = \{(x, y) : x \in R, y = f(x)\}$. Given functions f, g with $\mathrm{Graph}_R(f) \cap \mathrm{Grah}_R(g) = \emptyset$, the region of a cylinder bounded by two non-intersecting sections or bounded at one side is called *sector*; more precisely a *sector* is a region $S \subset \mathrm{Cyl}(R)$ having one of the following forms

$$S = \{(x, y) : x \in R, y < f(x)\}$$
$$S = \{(x, y) : x \in R, f(x) < y < g(x)\}$$
$$S = \{(x, y) : x \in R, g(x) < y\}$$

for some real continuous functions $f < g$ on R.

Definition 4.5. A *decomposition* of a set X is a finite collection $\mathcal{D} = \{X_1, ..., X_r\}$ of disjoint non-empty subsets (called *components*) whose union is X:

$$X = \bigcup_1^r X_i, \qquad X_i \cap X_j = \emptyset \text{ for } i \neq j$$

A finite collection of real continuous functions $f_1 < f_2 < ... < f_n$ on R yields a natural decomposition $\{R_0, R_1, ..., R_{2n}\}$ of $\mathrm{Cyl}(R)$ where:

$$\begin{aligned} R_0 &= \{(x, y) : x \in R, y < f_1(x)\} \\ R_1 &= \{(x, y) : x \in R, y = f_1(x)\} \\ R_2 &= \{(x, y) : x \in R, f_1(x) < y < f_2(x)\} \\ &\vdots \\ R_{2n-1} &= \{(x, y) : x \in R, y = f_n(x)\} \\ R_{2n} &= \{(x, y) : x \in R, f_n(x) < y\}. \end{aligned}$$

Such a decomposition is called a *stack over* R determined by $f_1, ..., f_n$. We can now introduce recursively the notion of cylindrical decomposition.

Definition 4.6. A decomposition $\mathcal{D}_d$ of $\mathbb{R}^d$ is said to be *cylindrical* if:

(i) when $d = 1$, $\mathcal{D}_d$ is a stack over $\mathbb{R}^0$, i.e. a decomposition of $\mathbb{R}$ into points and open intervals,

(ii) when $d > 1$, there is a cylindrical decomposition $\mathcal{D}_{d-1}$ of $\mathbb{R}^{d-1}$ and each region $R \in \mathcal{D}_{d-1}$ has a stack over R that is formed by regions of $\mathcal{D}_d$.

In this definition the decomposition $\mathcal{D}_{d-1}$ is unique for $\mathcal{D}_d$, hence it is clear that $\mathcal{D}_d$ induces uniquely the decompositions $\mathcal{D}_{d-1}, \mathcal{D}_{d-2}, ..., \mathcal{D}_1$ of the subspaces $\mathbb{R}^{d-1}, \mathbb{R}^{d-2}, ..., \mathbb{R}$.

Definition 4.7. We say that a decomposition is *algebraic* if its regions are semi-algebraic. The components (regions) of a cylindrical algebraic decomposition (CAD) are called *cells.*

Theorem 4.6 ([2]). Every cell of a cylindrical algebraic decomposition of $\mathbb{R}^d$ is semi-algebraically homeomorphic (i.e. the mapping is homeomorphic and semi-algebraic) to an open i-cube $(0,1)^i$, for some i, and is semi-algebraically connected.

Hence, we can naturally speak of the *dimension* of a cell; hence whenever we want to specify the dimension i, we say *i-cell.*

Let's consider a polynomial $f \in \mathbb{R}[x]$ in variables $x = (x_1, ..., x_d)$ and a semi-algebraic region $R \subset \mathbb{R}^{d-1}$. We call *folding point* (w.r.t coordinate x_d), a point $\hat{x}$ satisfying the system:

$$\begin{cases} f(\hat{x}) = 0 \\ \frac{\partial f}{\partial x_d}(\hat{x}) = 0 \end{cases}$$

Suppose that the cylinder $\mathrm{Cyl}(R)$ does not contain folding points. Then the equation

$$f(x) = 0$$

defines implicitly [19] the algebraic functions $g_1, ..., g_m : R \to \mathbb{R}$, such that $\mathrm{Graph}_R(g_i) \cap \mathrm{Graph}_R(g_j) = \emptyset$ if $i \neq j$. We can suppose:

$$g_1 < g_2 < ... < g_m$$

As a consequence, f induces a stack over R determined by functions that are implicitly definable by $f(x) = 0$. See Figure 4.2 for an illustration. We notice that in any region R_i of the stack over R the polynomial function $f(x)$ has constant sign, i.e. $\operatorname{sgn} f(x) = \operatorname{sgn} f(y)$ for all $x, y \in R_i$. In this case we also say that R_i is *f-sign-invariant* or f is *sign-invariant* on R_i.

Given the semi-algebraic region $R \subset \mathbb{R}^d$, consider now a collection $F = \{f_1, ..., f_n\}$ of polynomials in variables $x = (x_1, ..., x_d)$. We call *intersection point* of f_i and f_j $(1 \leq i < j \leq n)$ a point $\bar{x}$ satisfying the system:

$$\begin{cases} f_i(\bar{x}) = 0 \\ f_j(\bar{x}) = 0 \end{cases}$$

Suppose that the cylinder $\mathrm{Cyl}(R)$ does not contain folding points of f_i or intersection points of f_i, f_j, for all i, j s.t. $1 \leq i < j \leq n$. Then the graphs of the functions implicitly definable by $f_i = 0$ do not intersect the graphs of

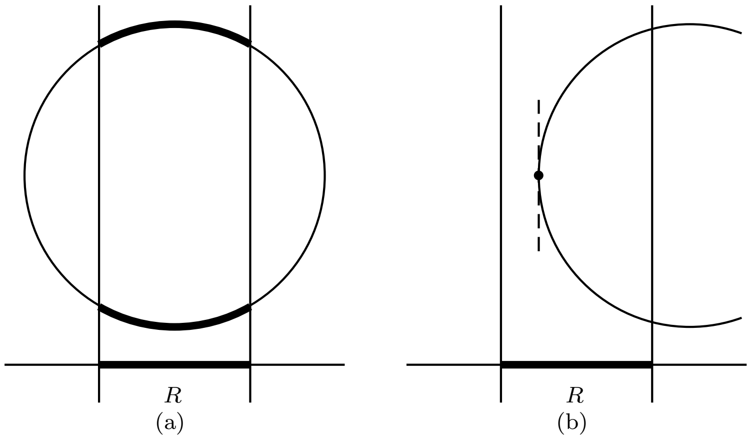

Fig. 4.2 (a) Stack over R; (b) No stack over R

the functions implicitly definable by $f_j = 0$: $F = \{f_1, ..., f_n\}$ induces a stack over R; see for instance Figure 4.3. Thus, we can conclude the following:

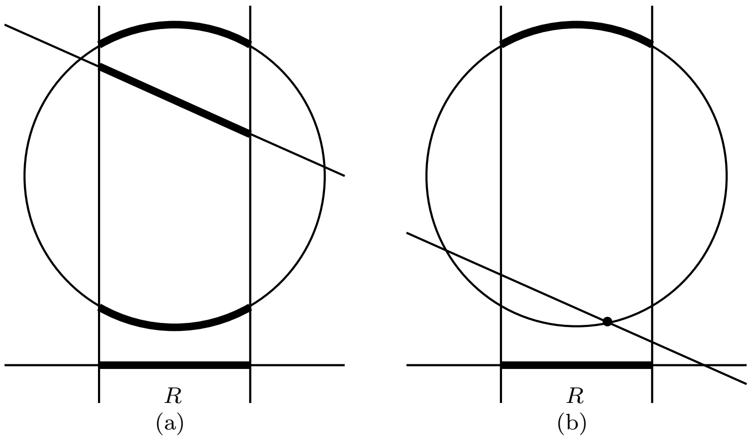

Fig. 4.3 (a) Stack over R; (b) No stack over R

Fact 4.7. In any region R_k of the stack over R every polynomial $f_i \in F$ has constant sign, i.e. $\operatorname{sgn} f(x) = \operatorname{sgn} f(y)$ for all $x, y \in R_k, f_i \in F$.

We say that the region R_k is *F-sign-invariant* or the collection F is *sign-invariant* on R_k.

Definition 4.8. A *cylindrical decomposition adapted to* $F \subset \mathbb{R}[x_1, ..., x_d]$ is a cylindrical decomposition whose cells are F-sign-invariant.

We will see now that for every finite collection F of polynomials the SCAD method effectively construct a cylindrical algebraic decomposition. To this aim we provide the insights into the SCAD algorithm. The algorithm SCAD takes a finite collection $F \subset \mathbb{Z}[x]$ of polynomials in the variables $x = (x_1, ..., x_d)$, and gives in output the points with dyadic components (i.e. components written as rational numbers with a denominator that is a power of 2) representing the sectors of a cylindrical algebraic decomposition of $\mathbb{R}^d$ adapted to F. The algorithm can be divided in three stages:

1. In the first stage (called *projection* or *elimination* stage) the last variable of the polynomials $F_d = F$ is eliminated, thus yielding a new collection $F_{d-1} \subset \mathbb{Z}[x_1, ..., x_{d-1}]$ of polynomials, in such a way that the following properties are guaranteed:

(a)For each distinct pair $f_i, f_j \in F_d$ there exists $g \in F_{d-1}$ such that, if $f_i(r_1, ..., r_d) = f_j(r_1, ..., r_d) = 0$, then $g(r_1, ..., r_{d-1}) = 0$.
(b)For each $f_i \in F_d$ there exists $h \in F_{d-1}$ such that, if $f_i(r_1, ..., r_d) = \frac{\partial}{\partial x_d} f_i(r_1, ..., r_d) = 0$, then $h(r_1, ..., r_{d-1}) = 0$.

An elimination procedure ELIM based on the concept of resultant will be discussed in Section 4.4.
By iterating the procedure ELIM on F_d, we obtain $F_{d-1}, F_{d-2}, ..., F_1$.

2. At this stage we consider the set F_1 of univariate polynomials, and let $q_1 < q_2 < ... < q_z$ be the (ordered) distinct real roots of the polynomials in F_1.
In this way we obtain a decomposition $\mathcal{D}$ of $\mathbb{R}$ adapted to F_1, whose sectors are the intervals:

$$I_0 = \{x : x < q_1\}, ..., I_j = \{x : q_j < x < q_{j+1}\}, ..., I_{z+1} = \{x : x > q_z\}$$

these sectors can be represented by dyadic numbers [9]:

$$d_0 < d_1 < ... < d_z < d_{z+1}$$

such that $d_k \in I_k$ for $0 \leq k \leq z + 1$. We call such numbers *sample points.*

The sample points $S_1 = \{d_0, ..., d_{z+1}\}$ of F_1 can be obtained by a procedure REP based on a bisection technique exploiting Sturm sequences, as explained in Section 4.6.

3. The lifting phase basically consists in finding the decomposition $\mathcal{D}_\ell$ of $\mathbb{R}^\ell$ adapted to F_ℓ given the polynomials in $F_{\ell-1}$ and the sample points of the decomposition $\mathcal{D}_{\ell-1}$ of $\mathbb{R}^{\ell-1}$ adapted to $F_{\ell-1}$. For this purpose we determine the stack over every $(\ell - 1)$-cell of the decomposition $D_{\ell-1}$ and we represent

the sectors of the stack by means of points with dyadic components. We recall that existence of such a stack is guaranteed as already observed in the projection phase.

Consider a sector $C \in \mathcal{D}_{\ell-1}$ represented by the sample point $s = (s_1, ..., s_{\ell-1}) \in S_{\ell-1} \cap C$. If we take the polynomials $f \in F_\ell \subset \mathbb{Z}[x_1, ..., x_\ell]$ and substitute the first $\ell - 1$ coordinates with s we obtain the set $F_\ell(s)$ of univariate polynomials $f_s(x_\ell) = f(s, x_\ell) \in \mathbb{Z}[x_\ell]$ in the variable x_ℓ. Now, in order to obtain the stack over s it is sufficient to apply the same technique of stage 2. on the polynomials $F_\ell(s)$. In conclusion, we can compute

$$\begin{aligned} S_\ell = \{ & s' = (s, s_\ell) \in \mathbb{Q}^\ell : s \in S_{\ell-1} \cap C, C \in \mathcal{D}_{\ell-1}, \\ & s_\ell \text{ is a sample point between consecutive roots of } F_\ell(s) \} \end{aligned}$$

The sign of each polynomial $f \in F_\ell$ in any cell $C \in \mathcal{D}_\ell$ having sample point $s' = (s, s_\ell)$ is easily determined by evaluating each polynomial $f_s \in F_\ell(s)$ at s_ℓ.

The construction of S_ℓ is iterated for $\ell = 2, ..., d$, yielding finally S_d, that are the points with dyadic components representing the sectors of the cylindrical algebraic decomposition of $\mathbb{R}^d$ adapted to F.

Remark 4.4. With an effective tool for performing a cylindrical algebraic decomposition one can also sketch the idea of a method [18] for solving the quantifier elimination problem stated in Section 4.2. We recall that it consists in finding a quantifier-free first-order formula Ψ that is equivalent to a given first-order formula Φ, possibly with quantifiers. The key point is that the realisation $\mathcal{R}_{\mathbb{R}}(\Phi) \subset \mathbb{R}^d$ of a first-order formula Φ constructed with the set of polynomials $F \subset \mathbb{Q}[x]$ is a union of some cells of a cylindrical decomposition $\mathcal{D}$ adapted to F [2]. Hence, it is sufficient to give a formula Ψ which is the disjunction of the polynomial equations and inequalities defining the cells of the decomposition $\mathcal{D}$.

Now we can formalise the Simplified Cylindrical Algebraic Decomposition in the Algorithm 5. The algorithm takes in input a finite collection of polynomials in t variables $\alpha_t, ..., \alpha_1$ and starts with eliminating repeatedly each variable by means of the ELIM$_\ell$ procedure outlined in Algorithm 6, which implements the resultant technique explained in Section 4.4, thus yielding the polynomial collections $F_t, ..., F_1$. In the second part (lines 4–11) the algorithm outputs all the sample points in the sectors of the decomposition $\mathcal{D}$ of $\mathbb{R}^t$. In this part the SCAD algorithm makes use of the REP procedure (Algorithm 8) implementing the techniques explained in Section 4.6 that generates sample points between the roots of the given univariate polynomial.
Observe that during the execution the algorithm generates into $S_1, ..., S_t$ also the sample points in the sectors of the decompositions of $\mathbb{R}, ..., \mathbb{R}^t$ as a result

Algorithm 5 SCAD

```
Input: $F = \{f_i \in \mathbb{Z}[\alpha_1, ..., \alpha_t]\}$
Output: a polynomial collection $S_t$ such that each sector of the cylindrical algebraic
    decomposition $\mathcal{D}$ of $\mathbb{R}^t$ has a sample point $\alpha \in \mathbb{Q}^t$ in $S_t$
1  $F_t = F$
2  for $\ell = t, ..., 2$ do
3      $F_{\ell-1} = \mathrm{ELIM}_\ell(F_\ell)$
4  $g = \prod_{f \in F_\ell} f$
5  $S_1 = \mathrm{REP}(g)$
6  for $\ell = 2, ..., t$ do
7      $g = \prod_{f \in F_l} f$                // $g \in \mathbb{Z}[\alpha_1, ..., \alpha_\ell]$
8      for all $s \in S_{\ell-1}$ do
9          $g_s(\alpha_\ell) = g(s; \alpha_\ell)$          // $g(s; \alpha_\ell) \in \mathbb{Z}[\alpha_\ell]$
10         $R = \mathrm{REP}(g_s)$
11         $S_\ell = S_\ell \cup \{(s, s_\ell) \in \mathbb{Q}^\ell : s_\ell \in R\}$
```

of the partial computations.

Since SCAD is a simplified version of Collin's CAD, its time complexity is surely bounded by CAD's time complexity. For our purpose it is then sufficient to illustrate CAD's complexity.

First of all, we give some bounds [2] on the complexity of the objects being computed throughout the algorithm. Let $F \subset \mathbb{Z}[x]$ be the collection of polynomials given in input. Denote by n the cardinality $|F|$, by p the maximum degree of the polynomials in F, by t the number of variables in the polynomials in F and by τ the maximum bitsize of the coefficients of the polynomials in F.

Proposition 4.8 ([2]). During the execution of the CAD algorithm the maximum degree of the computed polynomials is $O(p)^{2^{t-1}}$, the number of polynomials (i.e. cardinality of F_ℓ) is $O(np)^{3^{t-1}}$, and the bitsize of the computations is $\tau p^{O(1)^{t-1}}$.

Theorem 4.9 ([2]). Given a set $F \subset \mathbb{Z}[x]$ of n polynomials in t variables having maximum degree p and bitsize of coefficients bounded by τ, the Cylindrical Algebraic Decomposition algorithm gives a decomposition $\mathcal{D}$ adapted to F in time complexity $(np)^{O(1)^t}$. Moreover, the bitsizes of the computations and the output are bounded by $\tau p^{O(1)^{t-1}}$.

We first observe that although CAD's time complexity is doubly exponential in the number of variables, it remains polynomial in the input size and the integer p associated to the norm $\| \ \|_p$. In our application the number of variables will be fixed to $t = 2d + 1$, where d is the dimension of the data space.

Remark 4.5. Substantially, the construction of the $O(n^2)$ resultants associated to the pair of distinct polynomials in F_ℓ in each level $\ell = 1, ..., t$ is the reason for a doubly exponential (w.r.t. t) time complexity of the algorithm; indeed, at the second iteration of the projection phase the number of polynomials turns out to be $O(n^4)$, at the third iteration it turns to $O(n^8)$ and so on.

Proposition 4.10. Given the same polynomial set F of Theorem 4.9, the number of cells given by a Cylindrical Algebraic Decomposition $\mathcal{D}$ of $\mathbb{R}^t$ adapted to F is $(np)^{O(1)^t}$.

4.4 Resultant and elimination

We know from linear algebra that the solutions of a system of linear equations depend upon particular functions - namely rank and determinant - of the coefficients matrix. Similar techniques are then highly desirable also for systems of polynomial equations. As the Gaussian method can deal with eliminating variables in linear systems, some analogous approaches were developed to tackle the elimination in polynomial systems [7]. These approaches are based mainly on the Gröbner bases and the resultant. We will exploit the latter tool.

Let's first consider polynomials in one variable over the real field $\mathbb{R}$. We want to find the common roots of two polynomials a, b in $\mathbb{R}[x]$ of degree m and n respectively:

$$a(x) = a_m x^m + a_{m-1} x^{m-1} + ... + a_0 \tag{4.2}$$
$$b(x) = b_n x^n + b_{n-1} x^{n-1} + ... + b_0 \tag{4.3}$$

In particular, a simple way exists for deciding whether two polynomials have a common root, i.e. whether the system of two equations $a(x) = 0$ and $b(x) = 0$ admits a solution.

Definition 4.9 (Sylvester matrix). The *Sylvester matrix* $S(a, b)$ associated to the polynomials

$$a(x) = a_m x^m + ... + a_0 \in \mathbb{R}[x]$$
$$b(x) = b_n x^n + ... + b_0 \in \mathbb{R}[x]$$

of degree m and n respectively, is the $(m+n)$ by $(m+n)$ real matrix defined as follows:

$$S(a,b) = \left(\begin{array}{ccccccc}
a_m & a_{m-1} & \cdots & a_0 & & & \\
 & a_m & a_{m-1} & \cdots & a_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & a_m & a_{m-1} & \cdots & a_0 \\
\hline
b_n & b_{n-1} & \cdots & b_0 & & & \\
 & b_n & b_{n-1} & \cdots & b_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & b_n & b_{n-1} & \cdots & b_0
\end{array}\right)$$

the upper block having n rows, and the lower block having m rows, the blank entries filled by zeros .

In fact the Sylvester matrix is obtained by writing the coefficients vector of a on the upper left corner and then writing right-shifted copies on the $n-1$ rows below, and analogous operation for obtaining the lower block applies for the m successive rows.

Definition 4.10 (Resultant (Sylvester form)). The determinant $\det S(a,b)$ of the Sylvester matrix associated to a and b is called the *resultant* of a and b and is denoted by $\mathrm{Res}(a,b)$.

Remark 4.6. It is well known that the resultant is an integer polynomial in the a_i's and b_i's, and is homogeneous of degree n in the a_i's and homogeneous of degree m in the b_i's.

Furthermore, the following property holds:

Proposition 4.11. Denote by $\lambda_1, \lambda_2, ..., \lambda_m$ and by $\mu_1, \mu_2, ..., \mu_n$ the complex roots (counted with their multiplicity) of a and b respectively. Then

$$\mathrm{Res}(a,b) = a_m^n b_n^m \prod_{i,j} (\lambda_i - \mu_j)$$

The previous equality is often taken in literature as alternative definition of the resultant

Corollary 4.12. $\mathrm{Res}(a,b) = 0$ if and only if a and b have a common factor of degree > 0.

Hence, if we want to discover the solvability of the system of two polynomials in one unknown

$$\begin{cases} a(x) = 0 \\ b(x) = 0 \end{cases}$$

by the last Corollary it is then sufficient to recover the resultant of a and b.

Consider now two polynomials $a(x_1, ..., x_\ell), b(x_1, ..., x_\ell)$ in variables $x_1, ...,$ x_ℓ. We can interpret a and b as univariate polynomials in variable x_ℓ with coefficients in $\mathbb{R}[x_1, ..., x_{\ell-1}]$, i.e. $a, b \in \mathbb{R}[x_1, ..., x_{\ell-1}][x_\ell]$.

Their resultant is a polynomial $\mathrm{Res}(a,b)(x_1,...,x_{\ell-1})$ in variables $x_1,...,x_{\ell-1}$. Moreover, if $a(r_1,...,r_\ell)=b(r_1,...,r_\ell)=0$, then

$$\mathrm{Res}(a,b)(r_1,...,r_{\ell-1})=0.$$

This fact suggests the Algorithm 6 for the procedure of elimination. In fact,

Algorithm 6 ELIM

```
Input: a positive integer $\ell$; $F_\ell \subset \mathbb{Z}[\alpha_1,...,\alpha_\ell]$
Output: $F \subset \mathbb{Z}[\alpha_1,...,\alpha_{\ell-1}] : \beta \in \mathcal{Z}(F_\ell)$ is a folding, an intersection or a singular
    point only if $\pi(\beta) \in \mathcal{Z}(F)$
1  $F = \emptyset$
2  for all $f \in F_\ell$ do
3      $F = F \cup \mathrm{Res}(f, \frac{\partial f}{\partial \alpha_\ell})$          // singular or folding point w.r.t. $\alpha_\ell$
4      for all $g \in F_\ell, g \neq f$ do
5          $F = F \cup \mathrm{Res}(f,g)$          // intersection point w.r.t $\alpha_\ell$
```

for each distinct pair $f_i, f_j \in F_\ell$, there exists

$$g=\mathrm{Res}(f_i,f_j)\in F_{\ell-1}$$

such that, if $f_i(r_1,...,r_d)=f_j(r_1,...,r_d)=0$, then $g(r_1,...,r_{\ell-1})=0$; moreover for each f_i, if $f_i(r_1,...,r_\ell)=\frac{\partial}{\partial x_\ell}f_i(r_1,...,r_\ell)=0$ there exists

$$h=\mathrm{Res}(f_i,\frac{\partial}{\partial x_\ell}f_i)\in F_{\ell-1}$$

such that, if $f_i(r_1,...,r_d)=\frac{\partial}{\partial x_\ell}f_i(r_1,...,r_d)=0$ then $h(r_1,...,r_{d-1})=0$.

4.5 Sturm sequences

In this section we recall a classical tool that, given a polynomial, let us "localise" its roots.

Definition 4.11. Given $f,g\in\mathbb{R}[x]$ the *signed remainder sequence* of f and g is the finite sequence $\rho=(\rho_0,\rho_1,...,\rho_t)\subset\mathbb{R}[x]$ defined as follows:

$$\begin{aligned}
&\rho_0=f\\
&\rho_1=g\\
&\rho_2=-r_2 \text{ with the division } \rho_0=\rho_1 q_2+r_2\\
&\quad\vdots\\
&\rho_t=-r_t \text{ with the division } \rho_{t-2}=\rho_{t-1}q_t+r_t
\end{aligned}$$

with $\rho_t = -r_t = \mathrm{GCD}(f, g)$.

Essentially, this is a sign variated flavour of the Euclidean algorithm for computing the greatest common divisor.

Definition 4.12 (Sturm sequence). The *Sturm sequence* or *Sturm chain* of $f \in \mathbb{R}[x]$ is the signed remainder sequence of f and f', where f' is the derivative of f.

We denote by $V(\rho)$ the number of sign changes (ignoring zeros) in a given real sequence ρ, and given a Sturm sequence $\rho \subset \mathbb{R}[x]$ we denote by $V(\rho; a) = V(\rho(a))$ the number of sign changes (ignoring zeros) in the sequence $\rho_1(a), \rho_2(a), ..., \rho_t(a)$.

Theorem 4.13 (Sturm's Theorem [3]). Let $f \in \mathbb{R}[x]$ be a polynomial, $\rho = (\rho_0, ..., \rho_t)$ its Sturm sequence and $a < b$ two reals, which are not roots of f. Then the number of distinct roots of f in the open interval (a, b) is exactly

$$V(\rho; a) - V(\rho; b).$$

This result can be immediately turned into the procedure COUNTROOTS outlined in Algorithm 7. The algorithm takes in input a univariate polynomial g and the bounds a and b for the real roots of g. It depends on the function Rem which is a standard function for calculating the remainder of integer polynomial division. The lines 11-12 containing a nullity test guarantees that the output C is the number of real roots on the left-closed interval $[a, b)$.

Algorithm 7 COUNTROOTS(g, a, b)

```
Input: $g \in \mathbb{Z}[\alpha_l]; a, b \in \mathbb{Q}, a < b$
Output: the number $C$ of roots of $g$ in the interval $[a, b)$
1  $\rho_0 = g; \rho_1 = g'$
2  $i = 1$
3  while $\rho_i \neq 0$ do
4      $i = i + 1$
5      $\rho_i = -\mathrm{Rem}(\rho_{i-2}, \rho_{i-1})$        // -remainder of polynomial division
6  $\rho(a) = (\rho_0(a), \rho_1(a), ..., \rho_{i-1}(a))$
7  $\rho(b) = (\rho_0(b), \rho_1(b), ..., \rho_{i-1}(b))$
8  $U = \#$ sign changes in $\rho(a)$ ignoring zeros
9  $V = \#$ sign changes in $\rho(b)$ ignoring zeros
10 $C = U - V$
11 if $g(a) = 0$ then
12     $C = C + 1$
```

The previous theorem turns out to be an important tool for localising the real roots of a polynomial up to a desired approximation. Indeed, consider a polynomial $f \in \mathbb{R}[x]$ and an initial guess (a, b) for an interval containing all the roots. Suppose we want to localise one root of f. It is then sufficient to use a bisection approach; this means that we iteratively check the discending tower of intervals

$$(a,b) = (a_1,b_1) \supset (a_2,b_2) \supset (a_3,b_3) \supset ...$$

such that

$$V(\rho;a_i) - V(\rho;b_i) > 0$$

and (a_{i+1},b_{i+1}) is the left or right half-interval of (a_i,b_i), until we find the interval (a_k,b_k) containing exactly one root of f, namely $V(\rho;a_k)-V(\rho;b_i) = 1$. Obviously, it can happen that $f(a_i)=0$ or $f(b_i)=0$, which is not envisaged by the Sturm's Theorem; but it is sufficient to evaluate these points to test nullity.

Remark 4.7. When we want to sample a rational point between two consecutive roots $z_1 \in (a_1,b_1)$ and $z_2 \in (a_2,b_2), b_1 \le a_2$, we can give any rational number between b_1 and a_2, provided $b_1 < a_2$. Otherwise we have to refine the bisection on z_1 or z_2 until the latter condition is verified.

4.6 Cauchy's Bound and Canny's Gap

We now introduce a classical bound on the (real or complex) roots of a polynomial. This result turns to be very useful for giving the initial guess for the interval (a,b) containing all the real roots, and thus allowing one to apply a root finding method such as the bisection method we described above.

Lemma 4.14 (Cauchy's Bound on Roots [17]). Let $f(x) = a_n x^n + ... + a_0 \in \mathbb{C}[x]$ be a polynomial of degree n. Any root z of f lies in the disc:

$$|z| \le 1 + \max_i \left| \frac{a_i}{a_n} \right|$$

Corollary 4.15. If $f \in \mathbb{Z}[x]$ with coefficients of maximum bitsize τ, any root z of f lies in the disc

$$|z| \le 2^\tau$$

Proof. Simply note that $\left|\frac{a_i}{a_n}\right| \le |a_i| \le 2^\tau - 1$. □

With an initial guess due to the Cauchy's Bound we are ready to formalise the bisection method explained in the previous subsection into the Algorithm 8. At line 3 the algorithm guess the interval $(2^\tau, 2^{\tau+1})$ with right extreme $2^{\tau+1}$ since the procedure COUNTROOTS does not detect roots at the point 2^τ. The following lines consists in a in-order visit of a suitable tree $\mathcal{T}$ whose root is associated to the guess $(2^\tau, 2^{\tau+1})$. Actually, the recursive calls for the traversal of the tree are avoided by properly managing a stack of subintervals. Indeed, each node of $\mathcal{T}$ is associated to a subinterval $\mathcal{T}'$ of $(2^\tau, 2^{\tau+1})$. Whenever $\mathcal{T}'$ satisfies the two conditions:

(i) the left-half subinterval L of $\mathcal{T}'$ contains exactly one root
(ii) the right-half R contains no root

we can sample a rational point within the right-half R of $\mathcal{T}'$, since we are sure that the polynomial g does not change sign on R. Otherwise, we search in order L and R for some left and right-half subintervals satisfying the conditions (i) and (ii), provided they contain at least one root. We also notice that

Algorithm 8 REP(g)

```
Input: $g \in \mathbb{Z}[\alpha_l]$
Output: a list $S$ of sample points between consecutive roots of $g$
1  $S = \emptyset$
2  $\tau$ = maximum bitsize of $g$'s coefficients
3  $T = \{(-2^\tau, 2^{\tau+1})\}$        // $\tau + 1$: COUNTROOTS works on right open
4  while $T \neq \emptyset$ do
5      $(a, b) = \mathrm{Top}(T)$; $T = \mathrm{Pop}(T)$
6      $L = (a, \frac{a+b}{2})$
7      $R = (\frac{a+b}{2}, b)$
8      if COUNTROOTS($g, L$) $= 1$ and COUNTROOTS($g, R$) $= 0$ then
9          $S = S \cup \{\frac{1}{2}(\frac{a+b}{2} + b)\}$
10     else
11         if COUNTROOTS($g, R$) $\geq 1$ then
12             $T = \mathrm{Push}(T, R)$
13         if COUNTROOTS($g, L$) $\geq 1$ then
14             $T = \mathrm{Push}(T, L)$
```

the sample points generated into S are actually dyadic rational numbers, i.e. rationals with a power of 2 as denominator, as will be required in Section 4.8.

We've seen an upper bound on the absolute value of any root; on the contrary what follows can be directly applied to obtain a lower bound on the absolute value of any non-null root.

Theorem 4.16 (Canny's Gap [5]). Let $(x_1, x_2, ..., x_N)$ be a solution of an algebraic system of N equations in N unknowns having a finite number of solutions, with maximum degree d and with coefficients in $\mathbb{Z}$ smaller or equal to M in absolute value. Then, for each $i = 1, ..., N$, either $x_i = 0$ or $|x_i| > (3Md)^{-Nd^N}$.

Canny's Gap Theorem is a powerful tool for numerically solving symbolic decision problems. Indeed, this result can be exploited for deciding whether the solutions of a square system of algebraic equations are exactly zero or not, and hence in particular it can be used to test whether an algebraic number is zero by computing a sufficiently tight interval containing the number.

4.7 Comparison between two clusterings

In this section we develop a method that, having in input 2 clusterings of $X = \{x_1, ..., x_n\} \subset \mathbb{Z}^d$ with constraints $\{m, n-m\}$, decides which is better. The comparison is performed by formulating a proper algebraic system and using the Theorem 4.16 (Canny's Gap).

First of all, we extend the notion of ε–approximation, introduced in Chapter 1 for $\mathbb{R}$. Given a set $Y \subset \mathbb{R}^d$, consider each dimension i, $1 \leq i \leq d$. Let $y_1 < y_2 < ... < y_{m_i}$ be the distinct values of i-th component of the points in Y, and let j (that can depend on i) be the index such that the i-th component C_i of the p-centroid C of Y verifies

$$y_j \leq C_i < y_{j+1}$$

Fixed ε ($0 < \varepsilon < \frac{1}{2}$), we call *$\varepsilon$-approximation* of C the point $\bar{C} \in \mathbb{Q}^d$ having i-th component $\bar{C}_i$ satisfying:

$$\begin{cases} C_i \leq \bar{C}_i \leq C_i + \varepsilon & \text{if } y_{j+1} - C_i > C_i - y_j \\ C_i - \varepsilon \leq \bar{C}_i \leq C_i & \text{otherwise.} \end{cases}$$

In any case, it holds $|\bar{C}_i - C_i| \leq \varepsilon$ for every $i = 1, ..., d$.
The intuitive idea of these technicalities means that $\bar{C}_i$ must be either a left or a right approximation of C_i in such a way to ensure that $y_j \leq \bar{C}_i < y_{j+1}$. We notice that for each dimension i the corresponding j can be easily computed.

The following lemma is an easy extension of Proposition 1.7 introduced in Chapter 1 for $\mathbb{R}$, to the multi-dimensional case. It can be obtained by observing that the cost function W has no multivariate terms, so that the same analysis can be applied on each dimension separately.

Lemma 4.17. Given an integer $p > 2$ and the set $Y = \{y_1, ..., y_m\}$, let $\xi = \max \|y_i\|_\infty$ and $C \in \mathbb{R}^d$ be the p-centroid of Y and $W(Y)$ be the cost of the cluster Y. Then there exist polynomials $A_i(x_i) = \sum_{j=0}^{p-1} a_{ij} x_i^j$ ($i = 1, ..., d$) and $B(x_1, ..., x_d) = \sum_{i=1}^{d} \sum_{j=0}^{p} b_{ij} x_i^j$, such that:
1. C_i is a root of $A_i(x_i)$ for every $i = 1, ..., d$;
2. $W(Y) = B(C)$
3. $|a_{ij}|, |b_{ij}| \leq m \cdot (\xi + 1)^p$ for every i, j
4. $|B(C) - B(\bar{C})| \leq d \cdot \varepsilon \cdot \xi^{p-1} \cdot p \cdot m$.

We are now ready to establish the main result of this section.

Theorem 4.18. Given two 2-clusterings π_1 and π_2 of $X = \{x_1, ..., x_n\} \subset \mathbb{Z}^d$, the problem of deciding whether $W(\pi_1) < W(\pi_2)$ can be solved in polynomial time w.r.t. p and the bitsize of X.

Proof. We denote $\pi_1 = \{A, B\}$ and $\pi_2 = \{D, E\}$. Deciding whether $W(\pi_1) < W(\pi_2)$ corresponds to determining the sign of

$$W = W(\pi_1) - W(\pi_2) = W(A) + W(B) - W(D) - W(E)$$

To this end, we can consider the system of equations:

$$\begin{cases} A_i^{(k)}(z_i^{(k)}) = 0 \qquad \text{(for all } i = 1, ..., d; k = 1, ..., 4) \\ w = B^{(1)}(z^{(1)}) + B^{(2)}(z^{(2)}) - B^{(3)}(z^{(3)}) - B^{(4)}(z^{(4)}) \end{cases} \tag{4.4}$$

with polynomials $A_i^{(k)} \in \mathbb{R}[z_i^{(k)}], B^{(k)} \in \mathbb{R}[z^{(k)}]$ $(i = 1, ..., d; k = 1, ..., 4; z^{(k)} = (z_1^{(k)}, ..., z_d^{(k)}))$, obtained according to Lemma 4.17 separately for each cluster A, B, D, E and each term associated to a dimension in the cost function. This is a system of $4d + 1$ algebraic equations of degree at most p in $4d + 1$ unknowns $z^{(k)} \in \mathbb{R}^d$ $(k = 1, ..., 4)$ and w, which is solved by the assignments

$$\begin{cases} z^{(1)} = C^{(1)} \\ z^{(2)} = C^{(2)} \\ z^{(3)} = C^{(3)} \\ z^{(4)} = C^{(4)} \\ w = W \end{cases}$$

where the d-dimensional vectors $C^{(k)}$'s $(k = 1, ..., 4)$ are the (unique) centroids of A, B, D, E respectively. By Lemma 4.17, the coefficients of the polynomials in the system are bounded by $M = n(\xi + 1)^p$, where $\xi = \max_i \|x_i\|_\infty$. Hence, by applying the Canny's Gap Theorem, either $W = 0$ or $|W| > \delta$ with

$$\delta = [3n(\xi + 1)^p p]^{-(4d+1)p^{4d+1}}$$

Thus, if we find an approximation $\bar{W}$ of W up to $\frac{\delta}{3}$, we can conclude that:

- if $\bar{W} < -\frac{\delta}{2}$ then $W < 0$ and π_1 is the optimal solution;
- if $\bar{W} > \frac{\delta}{2}$ then $W > 0$ and π_2 is the optimal solution;
- if $|\bar{W}| \leq \frac{\delta}{2}$ then $W = 0$ and both π_1 and π_2 are optimal solutions.

$\bar{W}$ can be obtained by computing

$$\bar{W} = B^{(1)}(\overline{C^{(1)}}) + B^{(2)}(\overline{C^{(2)}}) - B^{(3)}(\overline{C^{(3)}}) - B^{(4)}(\overline{C^{(4)}})$$

where $\overline{C^{(k)}}$ is an ε-approximation of $C^{(k)}$, with ε that guarantees $|W - \bar{W}| \leq \frac{\delta}{3}$. By the last point of Lemma 4.17, we know that:

$$|W - \bar{W}| \leq 4d\varepsilon pn\xi^{p-1}.$$

To distinguish the three cases it is sufficient to choose ε such that

$$4d\varepsilon pn\xi^{p-1} < \frac{\delta}{3} = \frac{1}{3}[3n(\xi+1)^p p]^{-(4d+1)p^{4d+1}}.$$

Then we can approximate every component of $C^{(k)}$ up to the s-th binary digit after the point such that $\varepsilon = 2^{-s}$. By the previous inequality we have

$$\begin{aligned} 4d2^{-s}pn\xi^{p-1} &< \frac{1}{3}[3n(\xi+1)^p p]^{-(4d+1)p^{4d+1}} \\ -s &< \log[3n(\xi+1)^p p]^{-(4d+1)p^{4d+1}} - \log(12dpn\xi^{p-1}) \\ s &> (4d+1)p^{4d+1}\log[3n(\xi+1)^p p] + \log(12dpn\xi^{p-1}) \end{aligned}$$

Hence, noting that $\xi > n$, the minimum number of necessary binary digit is

$$s = O(p^{4d+2}\log\xi).$$

The approximate centroids $\overline{C^{(k)}}$ $(k = 1, ..., 4)$ can be obtained in polynomial time w.r.t. the input size and p by standard numerical methods for finding roots of polynomials, such as Bisection or Newton-Raphson Method, and the computation of $\bar{W}$ requires a polynomial number of arithmetic operations on numbers of polynomials size w.r.t. the input size and p.

In conclusion, we can decide in polynomial time w.r.t. p and the bitsize of X whether $W < 0$, that is whether $W(\pi_1) < W(\pi_2)$. □

The previous result suggests the Algorithm 9, which needs no further explanation.

Algorithm 9 IS_BETTER(π_1, π_2)

```
Input: two partitions $\pi_1 = \{A, B\}, \pi_2 = \{D, E\}$ of the data set $X \subset \mathbb{R}^d$; an even
    integer $p \geq 4$
Output: True if $W(\pi_1) < W(\pi_2)$, False otherwise
1  $\xi = \max_i \|x_i\|_\infty$                    // See Theorem 4.18
2  $\delta = [3n(\xi+1)^p p]^{-(4d+1)p^{4d+1}}$
3  $s = \lceil (4d+1)p^{4d+1}\log[3n(\xi+1)^p p] + \log(12dpn\xi^{p-1}) \rceil$
4  $\varepsilon = 2^{-s}$
5  Find $\varepsilon$-approximations $\overline{C^{(1)}}, \overline{C^{(2)}}, \overline{C^{(3)}}, \overline{C^{(4)}}$ of the centroids
   $C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}$ of $A, B, D, E$ respectively, up to the $s$-th digit by
   means of a standard numerical method.
6  $\bar{W} = B^{(1)}(\overline{C^{(1)}}) + B^{(2)}(\overline{C^{(2)}}) - B^{(3)}(\overline{C^{(3)}}) - B^{(4)}(\overline{C^{(4)}})$
7  if $\bar{W} < -\frac{\delta}{2}$ then
8      return True
9  else
10     return False
```

4.8 Size constrained 2-clusterings

4.8.1 Even p

We now return to the challange of finding the solution of the 2-SCC-d problem started in Section 4.1. Throughout this subsection we will suppose that p is even.

Take a point $x_i \in X$; we have already seen at the beginning of the chapter that it can be associated the polynomial introduced in Equation (4.1): $f_i(\alpha) = f(x_i; \alpha) \in \mathbb{R}[\alpha]$, namely

$$f_i(\alpha) = \sum_{k=1}^{d}(x_{i,k} - \mu_k)^p - \sum_{k=1}^{d}(x_{i,k} - \lambda_k)^p - \gamma$$

having collected the variables in the vector

$$\alpha = (\alpha_1, ..., \alpha_{2d+1}) = (\mu, \lambda, \gamma) = (\mu_1, ... \mu_d, \lambda_1, ..., \lambda_d, \gamma) \in \mathbb{R}^{2d+1}$$

Then let $F = \{f_i \in \mathbb{R}[\alpha] : x_i \in X\}$ be the collection of polynomials associated with the points of X, and $\mathcal{D}$ be the decomposition of $\mathbb{R}^{2d+1}$ adapted to F. Recall the Separation Property.

Fact 4.19. Let $\alpha \in \mathbb{R}^{2d+1}$ separate the clusters A and B. Then there exists a neighborhood $N_\varepsilon(\alpha) \in \mathbb{R}^{2d+1}$ of α such that

$$\alpha' \in N_\varepsilon(\alpha) \Longrightarrow \quad \text{sgn}(f_i(\alpha')) = \text{sgn}(f_i(\alpha)) \quad \forall i = 1, ..., n$$

This fact allows us to simplify the algorithm by considering only the sectors. Moreover, let's consider the *dyadic rational numbers*, that is those rationals of the form $\frac{a}{2^b}$ with integer a, and integer $b \geq 0$. As the dyadic numbers are dense in the reals, we can conclude

Fact 4.20. If $\alpha \in \mathbb{R}^{2d+1}$ separates the clusters A and B, then there exists a vector $\bar{\alpha} \in \mathbb{Q}^{2d+1}$ with dyadic rational components that separates A and B.

In order to find the clusterings separable by hypersurfaces of the parametric family

$$f(x; \mu, \lambda, \gamma) = \|x - \mu\|_p^p - \|x - \lambda\|_p^p - \gamma = 0$$

it is sufficient to apply the SCAD procedure of Section 4.3 to the set of polynomials:

$$F = \{f(x_i; \mu, \lambda, \gamma : i = 1, ..., n\}.$$

SCAD outputs a set of vectors

$$\{(\mu_k, \lambda_k, \gamma_k) : 1 \leq k \leq N\}$$

with dyadic components. By the Separation Property, we know that, for every $1 \leq m \leq n-1$, there is $j, 1 \leq j \leq N$, such that the optimal clustering with constraints $\{m, n-m\}$ is obtained by the hypersurface $f(x; \mu_j, \lambda_j, \gamma_j) = 0$. In conclusion, fixed a norm $\| \; \|_p$ with even p, given a set $\{x_1, ..., x_m\} \subset \mathbb{R}^d$ of vectors, the optimal clusterings with constraints $\{m, n-m\}$ (for all $1 \leq m \leq n-1$) can be obtained by the Algorithm 10.

Algorithm 10 2-SCC-d

```
Input: an even integer $p \geq 4$; $X = \{x_i\}_1^n \subset \mathbb{Z}^d$
Output: a vector $\Pi = (\pi[1], ..., \pi[\lfloor n/2 \rfloor])$, where $\pi[m]$ is the solution to the 2-SCC-$d$
    problem with constraint $m$, for every $m = 1, ..., \lfloor n/2 \rfloor$.
1  $F_{2d+1} = \{f \in \mathbb{Z}[\alpha] : f(x; \mu, \lambda, \gamma) = \|x-\mu\|_p^p - \|x-\lambda\|_p^p - \gamma, x \in X\}$
2  $S_{2d+1} = \mathrm{SCAD}(F_{2d+1})$
3  for all $\alpha \in S_{2d+1}$ do
4      $m = 0; A = \emptyset$
5      for all $x_i \in X$ do
6          if $f(x_i; \alpha) > 0$ then
7              $A = A \cup \{x_i\}$
8              $m = m + 1$
9      $m = \min\{m, n-m\}$
10     $\pi = \{A, X \setminus A\}$
11     if IS_BETTER($\pi[m], \pi$) then
12         $\pi[m] = \pi$
```

Complexity analysis

We now want to determine the time complexity of the 2-SCC-d algorithm. We recall from Section 4.3 that the SCAD algorithm is polynomial-time w.r.t. to input size and p. Because of Proposition 4.10 the number of iterations in the **for** loop at line 3 is $(np)^{O(1)^{2d+1}}$. Each evaluation of the polynomial function $f(x_i; \cdot)$ at line 6 requires $5d-1$ additions or subtractions and $2d$ exponentations in the ring $\mathbb{R}$. As stated by the Theorem 4.18 the time complexity of the IS_BETTER function is polynomial w.r.t. to p and the bitsize of X. It results that the cost of the whole **for** loop of lines 3–12 is polynomial in p and the bitsize of X. From these considerations the following result holds.

Theorem 4.21. The 2-clustering problem with size constraints k, $1 \leq k \leq \lfloor n/2 \rfloor$, in fixed dimension d with norm $\| \; \|_p$, even integer p, can be solved in polynomial time w.r.t. the input size and p.

4.8.2 Odd p

We recall that, as in Section 4.1, with each point $x_i \in X$ we can associate the function

$$f_i(\alpha) = \sum_{\ell=1}^{d} |x_{i\ell} - \mu_\ell|^p - |x_{i\ell} - \lambda_\ell|^p - \gamma$$

denoting with $\alpha = (\mu, \lambda, \gamma) \in \mathbb{R}^{2d+1}$. When p is a positive odd number, the considerations of §4.8.1 can no longer be applied directly. In this subsection we want to illustrate how the case of odd p can be deal with in an analogous manner with little complication.

Let $F = \{f_i \in \mathbb{R}[\alpha] : x_i \in X\}$; by reasoning as in §4.8.1 we know that for each α the sign vector

$$\operatorname{sgn}(F(\alpha)) = (\operatorname{sgn}(f_1(\alpha)), ..., \operatorname{sgn}(f_n(\alpha)) \in \{-1, +1\}^n$$

of the polynomials in F evaluated at α determines the unique 2-clustering π_α.

We now introduce some modified version of the functions in F. Let $\sigma, \tau \in \{-1, +1\}^{n\times d}$ denote sign matrices and let $\sigma_i, \tau_i,\ i = 1, ..., n$, denote their respective rows. To every sign vectors $\sigma_i, \tau_i \in \{-1, +1\}^d$ we associate the polynomial

$$\bar{f}_{i\sigma_i\tau_i}(\alpha) = \sum_{\ell=1}^{d} \sigma_{i\ell}(x_{i\ell} - \mu_\ell)^p - \tau_{i\ell}(x_{i\ell} - \lambda_\ell)^p - \gamma$$

where $\sigma_{i\ell}, \tau_{i\ell}$ are entries of σ_i, τ_i, and to each point $x_i \in X$ we associate the polynomial set

$$\Psi_i = \{\bar{f}_{i\sigma_i\tau_i} \in \mathbb{R}[\alpha] : \sigma_i, \tau_i \in \{-1, +1\}^d\}$$

containing $|\Psi_i| = 4^d$ polynomials. We can then construct the polynomial set

$$\bar{F} = \bigcup_{1}^{n} \Psi_i \tag{4.5}$$

containing $|\bar{F}| = n4^d$ polynomials. Define also the polynomial set

$$\begin{aligned} G = \{(x_{i\ell} - \mu_\ell) \in \mathbb{R}[\alpha] : i = 1, ..., n; \ell = 1, ..., d\} \\ \cup \{(x_{i\ell} - \lambda_\ell) \in \mathbb{R}[\alpha] : i = 1, ..., n; \ell = 1, ..., d\} \end{aligned} \tag{4.6}$$

containing $|G| = 2nd$ polynomials.

Proposition 4.22. Let $\mathcal{D}$ be the cylindrical decomposition of $\mathbb{R}^{2d+1}$ adapted to $H = \bar{F} \cup G$. Then $\mathcal{D}$ is also a cylindrical decomposition of $\mathbb{R}^{2d+1}$ adapted to F.

Proof. Take an arbitrary cell $C \in \mathcal{D}$ and two points $\alpha' = (\mu', \lambda', \gamma'), \alpha'' = (\mu'', \lambda'', \gamma'')$ in C. Let's denote the $(2d+1)$–dimensional variable $\alpha = (\mu, \lambda, \gamma)$.

The polynomials $(x_{i\ell} - \mu_\ell), (x_{i\ell} - \lambda_\ell)$ belong to H for all $i = 1, ..., n; \ell = 1, ..., d$. Hence, since $\mathcal{D}$ is adapted to H, for every $x_i \in X$ there are suitable $\xi, \zeta \in \{-1, +1\}^n$ such that

$$\begin{aligned} \operatorname{sgn}(x_{i\ell} - \mu'_\ell) = \operatorname{sgn}(x_{i\ell} - \mu''_\ell) = \xi_\ell \qquad \ell = 1, ..., d \\ \operatorname{sgn}(x_{i\ell} - \lambda'_\ell) = \operatorname{sgn}(x_{i\ell} - \lambda''_\ell) = \zeta_\ell \qquad \ell = 1, ..., d \end{aligned}$$

It follows immediately that

$$\sum_{\ell=1}^{d} |x_{i\ell} - \mu'_\ell|^p - |x_{i\ell} - \lambda'_\ell|^p - \gamma' = \sum_{\ell=1}^{d} \xi_\ell (x_{i\ell} - \mu'_\ell)^p - \zeta_\ell (x_{i\ell} - \lambda'_\ell)^p - \gamma' \quad (4.7)$$

and

$$\sum_{\ell=1}^{d} |x_{i\ell} - \mu''_\ell|^p - |x_{i\ell} - \lambda''_\ell|^p - \gamma'' = \sum_{\ell=1}^{d} \xi_\ell (x_{i\ell} - \mu''_\ell)^p - \zeta_\ell (x_{i\ell} - \lambda''_\ell)^p - \gamma''. \quad (4.8)$$

By construction we know that for any $f_i \in F$ there is $\bar{f}_{i\xi\zeta} \in \bar{F}$, and hence it follows $f_i(\alpha') = \bar{f}_{i\xi\zeta}(\alpha')$ because of (4.7) and $f_i(\alpha'') = \bar{f}_{i\xi\zeta}(\alpha'')$ because of (4.8). However $\operatorname{sgn}(\bar{f}_{i\xi\zeta}(\alpha')) = \operatorname{sgn}(\bar{f}_{i\xi\zeta}(\alpha''))$ since, by the hypothesis, C is $\bar{f}_{i\xi\zeta}$-sign-invariant.

We can conclude that $\operatorname{sgn}(f_i(\alpha')) = \operatorname{sgn}(f_i(\alpha''))$ for all $i = 1, ..., n$, hence C is F-sign-invariant and the claim follows. □

With this result we can apply the same technique used in the case p even. Starting from collection F we construct $H = \bar{F} \cup G$. We execute the SCAD algorithm on H, thus obtaining the dyadic sample points α in the F-sign-invariant cells, and then exploit the standard numerical method for finding zeros in order to compare all the 2-clusterings π_α associated to the sample points α.

Theorem 4.23. The size constrained 2-clustering problem in fixed dimension d with norm $\| \|_p$, odd integer p, can be solved in polynomial time w.r.t. the input size and p.

Proof. Let $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ be the set of points in input. Apply a modified version of Algorithm 10 which taking in input an odd integer $p \geq 3$ instead of an even p, starts by constructing in polynomial time the collections $\bar{F}$ and G as specified in Equations (4.5) and (4.6), and then executes the modified version of line 1:

$$1' \quad F_{2d+1} = \bar{F} \cup G$$

The algorithm then continues with line 2 without other modifications.
By construction of $\bar{F}$ and G it follows that the set F_{2d+1} at line $1'$ contains

$|F_{2d+1}| = n4^d + 2nd$ polynomials with coefficients having the same bitsize of the polynomials in F. Hence the CAD decomposition of $\mathbb{R}^{2d+1}$ adapted to F_{2d+1} still yields a polynomial number of cells in polynomial time (w.r.t. p and the input size), thus giving in S_{2d+1} a polynomial number of sample points. The complexity analysis of the remainder of the algorithm is the same of §4.8.1. $\square$

References

1. D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical Algebraic Decomposition I: The Basic Algorithm. Technical Report 351, Department of Computer Science, Purdue University, 1982.
2. S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry.* Springer, Berlin, 2003.
3. J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry.* Springer, Berlin, 1998.
4. J. Canny. *The complexity of robot motion planning.* MIT Press, Cambridge, MA, USA, 1988.
5. J. Canny. Some algebraic and geometric computations in PSPACE. In *ACM Symposium on the Theory of Computation*, pages 460–367, 1988.
6. P. J. Cohen. Decision Procedures for Real and p-adic Fields. *Comm. Pure and Applied Math.*, 22(2):131–151, March 1969.
7. P. M. Cohn. *Classic Algebra.* Wiley, 3rd edition, 2000.
8. G. E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In E. Barkhage, editor, *Proc. 2nd GI Conf. on Automata Theory and Formal Lang.*, volume 33 of *LNCS*, pages 134–183, Berlin, 1975. Springer.
9. G. E. Collins, J. R. Johnson, and W. Krandick. Interval arithmetic in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 34(2):145 –157, 2002.
10. M. Coste. Effective semi-algebraic geometry. *Lect. Notes in Comp. Sci.*, 391:1–27, 1989.
11. D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer, New York, 3rd edition, 2007.
12. D. Geiger and C. Meek. Quantifier Elimination for Statistical Problems. In K. Laskey and H. Prade, editors, *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, pages 226–235. AUAI, Morgan Kaufmann, July 1999.
13. J. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry.* Discrete mathematics and its applications. Chapman & Hall/CRC, 2004.
14. D. Grigor'ev and N. Vorobjov. Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5:37–64, 1988.
15. R. Hartshorne. *Algebraic Geometry.* Springer, New York, 1977.
16. J. Heintz, M.-F. Roy, and P. Solernò. On the theoretical and practical complexity of the existential theory of reals. *The Computer Journal*, 36(5):427–431, 1993.
17. V. Jain. On Cauchy's bound for zeros of a polynomial. *Approximation Theory and its Applications*, 6:18–24, 1990.
18. S. LaValle. *Planning algorithms.* Cambridge University Pres, 2006.
19. W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, 3rd edition, 1976.
20. A. Seidenberg. A New Decision Method for Elementary Algebra. *Annals of Math.*, 60(2):365–374, Sept. 1954.
21. I. R. Shafarevich. *Basic Algebraic Geometry 2.* Springer, Berlin, 1977.
22. J. H. Silverman. *The Arithmetic of Elliptic Curves.* Springer, New York, 2009.
23. A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, Berkeley, USA, 2nd rev. edition, 1951.

Conclusions

In this work we have studied some algorithmic problems on distance clustering with size constraints in the space $\mathbb{R}^d$ endowed with L^p-norm. In particular, we have obtained separation results for the optimal solutions that, in the 1-dimensional case, imply the so-called String Property. In this way a well-known result in clustering is extended to constrained clustering. Moreover, we have also introduced a relaxed version of the constrained clustering problem, where the size of the clusters can vary within a given set. It turns out that this relaxed version is a generalisation of the classical clustering problem.

We showed that the constrained clustering problem is difficult in general. In fact, we proved that even fixing the number k of clusters, the problem is NP-hard. Moreover, by taking an integer parameter p of the norm the decision version of the Half Partition problem is NP-complete, while we have given evidence that, on the contrary, the methods cannot be extended to the case of rational non-integer p. Furthermore, we have shown that constrained clustering is NP-hard even in dimension 1, while the corresponding problem in classical clustering is solvable in polynomial time, at least with the Euclidean norm.

By a non-classical reduction from the Planar 3-SAT problem we have also shown that the decision version of the relaxed constraints clustering is NP-complete.

The problem seems to be easier by fixing both dimension d and number k of clusters. In particular, in the planar case with Euclidean norm, we built an efficient algorithm for finding all the solutions of the constrained 2-clustering problems with every cluster size $m = 1, ..., \lfloor n/2 \rfloor$. The case of 2-clustering is particularly interesting since it is the base step in the family of divisive hierarchical clustering techniques. Moreover, by relying on some results of the combinatorial geometry about the k-sets counting question and on some well-designed data structures for handling the convex hulls, we have also found an efficient algorithm for constructing one solution of the constrained 2-

clustering problem with a given cluster size in the planar case with Euclidean norm.

It remains to extend some techniques used in the Euclidean case to the case of Manhattan norm. In this regard, we conjecture that some efficient tree structures can be used to solve the problem efficiently. Moreover, it is natural to extend the results of the planar case to the multi-dimensional case; this extension seems to be direct for the 2-clustering with Euclidean norm since the clusters are separated by hyperplanes.

When both the dimension d and the number k of clusters are fixed, and the integer parameter p of the norm is given in unary representation, we have shown that the constrained 2-clustering problem is solvable in polynomial time. This result is mainly based on a real algebraic geometry method, namely the so-called cylindrical algebraic decomposition of the parameter space of the separating hypersurfaces and it also relies on some numerical techniques for localising real algebraic roots.

The methods applied in this case depend on the fact that $\| \cdot \|_p^p$ is a semi-algebraic function. An open problem is the simplification of these methods by exploiting the very particular semi-algebraic form of $\| \cdot \|_p^p$.