Exact Algorithms for 2-Clustering with Size Constraints in the Euclidean Plane*

Alberto Bertoni, Massimiliano Goldwurm, and Jianyi Lin

Dipartimento di Informatica, Università degli Studi di Milano,
Via Comelico 39/41, 20135 Milano, Italy
jianyi.lin@unimi.it

Abstract. We study the problem of determining an optimal bipartition $\{A, B\}$ of a set X of n points in $\mathbb{R}^2$ that minimizes the sum of the sample variances of A and B, under the size constraints $|A| = k$ and $|B| = n-k$. We present two algorithms for such a problem. The first one computes the solution in $O(n\sqrt[3]{k}\log^2 n)$ time by using known results on convex-hulls and k-sets. The second algorithm, for an input $X \subset \mathbb{R}^2$ of size n, solves the problem for all $k = 1, 2, \ldots, \lfloor n/2 \rfloor$ and works in $O(n^2 \log n)$ time.

Keywords: algorithms for clustering, cluster size constraints, data analysis, Euclidean distance, machine learning.

1 Introduction

The general Clustering Problem consists in finding an optimal partition of a set X of n points in m clusters, i.e. a partition of X in m subsets that minimizes the sum of the dispersion of points around the centroid in each subset. This is a fundamental problem in many research areas like data mining, image analysis, pattern recognition and bioinformatics [18]. Clustering is a classical method in unsupervised machine learning, frequently used in statistical data analysis [5,10].

A computational analysis of the problem depends on a variety of parameters: the dimension d of the point space (usually $\mathbb{R}^d$), the distance or semi-distance used to measure the dispersion of points, the number m of clusters (which may be arbitrary, as part of the instance, or fixed in advance), the size of the clusters and possibly others constraints [2,20,21]. In most cases the problem is difficult. For instance, assuming the squared Euclidean semi-distance, when the dimension d is arbitrary the general Clustering Problem is NP-hard even if the number m of clusters is fixed to 2 [1,6]. The same occurs if m is arbitrary and the dimension is $d = 2$ [14]. The problem is solvable in polynomial time when fixing both m and d [11]. Moreover there exists a well-known, usually fast, heuristic for finding an approximate solution, called k-Means [13], which however requires exponential time in the worst case [19].

* This research has been supported by project PRIN #H41J12000190001 "Automata and formal languages: mathematical and applicative aspects".

G.F. Italiano et al. (Eds.): SOFSEM 2015, LNCS 8939, pp. 128–139, 2015.

Thus, a natural goal of research in this context is to study particular cases with suitable hypothesis on the input that allow us to design polynomial time algorithms. Here, we consider the Clustering Problem in $\mathbb{R}^2$, assuming squared Euclidean semi-distance, when the number of clusters is $m = 2$ and their size is given by the instance. We call it *Size Constrained 2-Clustering* in $\mathbb{R}^2$ (2-SCC-2 for short).

More precisely, an instance of this problem is given by a set $X \subset \mathbb{R}^2$ of n points in general position and an integer k such that $1 \leq k \leq n/2$, while the solution is a bipartition $\{A, B\}$ of X such that $|A| = k$, that minimizes the total weight $W(A) + W(B)$, where $W(A)$ (respectively, $W(B)$) is the sum of the squares of the ℓ_2-distances of all points $a \in A$ (resp. $b \in B$) from the centroid of A (resp. B). A more formal description is given in Section 3. Recall that the unconstrained version of the same problem, with an arbitrary number of clusters, is NP-hard [14].

The relevance of the 2-clustering problems is due to the wide spread of hierarchical clustering techniques, that repeatedly apply the 2-clustering as the key step. The 2-clustering problem with cluster size constraints has been already studied in [12,4], where it is shown that in dimension 1 the problem is solvable in polynomial time for every norm ℓ_p with integer $p \geq 1$, while there is some evidence that the same result does not hold for non-integer p. It is also known that for arbitrary dimension d the same problem is NP-hard even assuming equal sizes of the two clusters.

In this work we show two results. First, we describe an algorithm that solves 2-SCC-2 in $O(n\sqrt[3]{k}\log^2 n)$ time. This is obtained by using known results of computational geometry concerning in particular dynamic data structures for convex hulls [17,16] and the enumeration of k-sets in $\mathbb{R}^2$ [9,7]. Then, we present an algorithm for the full-version of the problem, i.e. a procedure yielding a solution for all $k = 1, 2, \ldots, \lfloor n/2 \rfloor$, which works in $O(n^2 \log n)$ time. Both algorithms are based on a separation result on the clusters of optimal solutions for 2-SCC-2, presented in Section 3 and proved in [4], which intuitively extends to the bidimensional case the so-called String Property of the optimal clusterings on the real line [15]. The results we present here are obtained by assuming the Euclidean norm and this hypothesis is crucial our proofs. We observe that the Euclidean norm is an essential hypothesis in our proofs and the results we present here do not seem to hold under different assumptions. For instance, in the case of Manhattan distance (ℓ_1 norm), the separation result on the plane yields a $O(n^3 \log n)$ time algorithm for the full-version of the problem [3].

2 Preliminary Notions

In this section we fix our notation and recall some known results of computational geometry [8,17].

For any point $a \in \mathbb{R}^2$, we denote by a_x and a_y the abscissa and the ordinate of a, respectively. We denote by $\|a\|$ the usual Euclidean norm of point a, i.e. $\|a\| = (a_x^2 + a_y^2)^{1/2}$. We also fix a total order on points in $\mathbb{R}^2$: for every $a, b \in \mathbb{R}^2$,

we set $a <_o b$ if either $a_y < b_y$ or $a_y = b_y \wedge a_x < b_x$. Clearly, every point a defines a vector of length $\|a\|$ oriented from the origin to a.

Given two points $a, b \in \mathbb{R}^2$, the oriented line segment from a to b is called *oriented edge* and is identified by the pair (a, b). By a little abuse of language we also denote by (a, b) the *straight line* through the two points oriented from a to b. We define the (positive) *phase* of (a, b) as the angle between the oriented edges $(a, (a_x + 1, a_y))$ and (a, b), measured counter-clockwise. We denote it by $phase(a, b)$. Clearly, we have $0 \leq phase(a, b) < 2\pi$. Note that two oriented edges have the same phase if and only if they are parallel and have the same orientation.

We also define the *slope* of (a, b) as the remainder of the division $phase(a, b)/\pi$ and we denote it by $slope(a, b)$. Observe that (a, b) and (b, a) have the same slope and, more generally, two oriented edges have the same slope if and only if they are parallel (with either equal or opposite orientation).

Moreover, for every pair of oriented edges (a, b), (c, d), we say that (a, b) is *on the right* of (c, d) and write $(a, b)\text{Right}(c, d)$ if the position of the straight line (c, d) is obtained by a counter-clockwise rotation of the straight line (a, b) (around their intersection point) smaller than π. On the contrary, if such a rotation is greater than π we say that (a, b) is *on the left* of (c, d) and write $(a, b)\text{Left}(c, d)$. Note that relations *Right* and *Left* correspond respectively to the positive and negative sign of the cross product $(a, b) \times (c, d)$, which is determined by the well-known right-hand rule.

Clearly, once the coordinates of points are known, one can compute in constant time both phase and slope of a point, as well as establish the validity of relations *Right* and *Left* between two oriented edges.

Now, let us consider a finite set $X \subset \mathbb{R}^2$: we say that X is in *general position* if it does not contain 3 collinear points. Moreover, if X consists of n points, for any integer $0 \leq k \leq n$ a *k-set* of X is a subset $A \subseteq X$ of cardinality $|A| = k$ such that $A = X \cap H$ for a suitable half-space $H \subset \mathbb{R}^2$. This means A is separable from its complement $\bar{A}$ by a straight-line. Determining the maximum number of k-sets of a family of n points in $\mathbb{R}^2$ is a central problem in combinatorial geometry, first posed in [9].

Recall that the intersection of an arbitrary collection of convex sets is convex. Then for any set $A \subseteq \mathbb{R}^d$, the *convex hull* or *convex closure* of A, denoted by $\text{Conv}(A)$, is defined as the smallest convex subset of $\mathbb{R}^d$ containing A, i.e.

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

It is well-known that the convex closure of a finite set A of points in $\mathbb{R}^d$ is a polytope [17] determined by the intersection of finitely many half-spaces; in particular in $\mathbb{R}^2$, $\text{Conv}(A)$ is a convex polygon. It is possible to identify a polygon by giving its vertices, and hence the determination of the convex closure $\text{Conv}(A)$ of a given set $A \subset \mathbb{R}^2$ consists in finding the vertices of the associated polygon. We recall that the convex hull of a set X of n points in $\mathbb{R}^2$ can be computed in time $O(n \log n)$ [17].

3 Problem Definition and First Properties

To give a formal definition of the problem, we recall that a *cluster* of a finite set $X \subset \mathbb{R}^2$ is a non-empty subset $A \subset X$, while the pair $\{A, \bar{A}\}$ is a *2-clustering* of X, where $\bar{A} = X \setminus A$ is the complement of A. Assuming the Euclidean norm $\|\cdot\|$, the *centroid* of A is the value $C_A \in \mathbb{R}^2$ defined by

$$C_A = \operatorname*{argmin}_{\mu \in \mathbb{R}^2} \sum_{a \in A} \|a - \mu\|^2.$$

It turns out that C_A is the mean value of points in A: $C_A = \frac{\sum_{a \in A} a}{|A|}$. Moreover, we denote by $W(A)$ the *weight* of A, that is

$$W(A) = \sum_{a \in A} \|a - C_A\|^2$$

Note that $\frac{W(A)}{|A|-1}$ is the traditional sample variance of A, once we interpret the elements of A as sample points picked up from a random variable in $\mathbb{R}^2$ (rather than in $\mathbb{R}$). Hence, it represents a natural measure of the dispersion of points in A around their mean value.

Then, the 2-SCC-2 problem is defined as follows:

> Given a set $X \subset \mathbb{R}^2$ of cardinality n (in general position) and an integer k, $1 \leq k \leq n/2$, find a 2-clustering $\{A, \bar{A}\}$ of X, with $|A| = k$, that minimizes the weight $W(A, \bar{A}) = W(A) + W(\bar{A})$.

By the observation above, since the size of the clusters is constrained, this is equivalent to looking for the 2-clustering (A, B) that minimizes the sum of the sample variances of A and B.

The weight of a 2-clustering in the plane can be computed using the following proposition, the proof of which is here omitted for sake of brevity.

Proposition 1. *Let $\{A, B\}$ be a 2-clustering of a set $X \subset \mathbb{R}^2$ of n points such that $|A| = k$ for some $k \in \{1, 2, \ldots, \lfloor n/2 \rfloor\}$. Then*

$$W(A, B) = \sum_{p \in X} \|p\|^2 - \frac{n}{k(n-k)} \|S_A\|^2$$

where $S_A = \sum_{a \in A} a$.

As a consequence, solving the 2-SCC-2 problem for an instance (X, k) is equivalent to determining a subset $A \subseteq X$ of size k that maximizes the value $\|S_A\|$.

Another property we use in the present work is the following separation result between the clusters of optimal solutions of the 2-SCC-2 problem, proved in [4].

Proposition 2 (Separation Result). *Let $\{A, B\}$ be an optimal solution of the 2-SCC-2 problem for an instance $X \subset \mathbb{R}^2$ with constraint $|A| = k$. Then, there exists a constant $c \in \mathbb{R}$ such that, for every $p \in X$,*

$$\begin{aligned} p \in A \quad &\Rightarrow \quad 2p_x(C_{Bx} - C_{Ax}) + 2p_y(C_{By} - C_{Ay}) \;<\; c + \|C_B\|^2 - \|C_A\|^2, \\ p \in B \quad &\Rightarrow \quad 2p_x(C_{Bx} - C_{Ax}) + 2p_y(C_{By} - C_{Ay}) \;>\; c + \|C_B\|^2 - \|C_A\|^2. \end{aligned}$$

As a consequence, both clusters of any optimal solution $\{A, B\}$ of the 2-SCC-2 problem are separated by the straight line of equation

$$2x(C_{Bx} - C_{Ax}) + 2y(C_{By} - C_{Ay}) = c + \|C_B\|^2 - \|C_A\|^2$$

for some constant $c \in \mathbb{R}$. This implies that A is a k-set and B is a $(n-k)$-set.

This result can be interpreted as a natural extension of the well-known String Property, stating that all optimal clusterings on the real line consist of contiguous subsets of the input set (see for instance [15]).

Moreover, Propositions 1 and 2 imply that the 2-SCC-2 problem for an instance (X, k) can be solved by computing a k-set $A \subseteq X$ with maximum $\|S_A\|$.

4 Algorithm for Constrained 2-Clustering in $\mathbb{R}^2$

In this section we present an efficient technique, based on Proposition 2, to solve the 2-SCC-2 problem as defined in Section 3. Here the input is given by a set $X \subset \mathbb{R}^2$ of $n \geq 4$ points in general position and an integer k such that $1 < k \leq \lfloor n/2 \rfloor$. Our purpose is to show that the algorithm works in $O(n\sqrt[3]{k}\log^2 n)$ time.

We first introduce some preliminary notions. Given two disjoint polygons it is easy to see that there exist 4 straight lines that are tangent to both polygons: they 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 define

$$X^r(a,b) = \{p \in X \mid (a,p)\text{Right}(a,b)\},$$
$$X^l(a,b) = \{p \in X \mid (a,p)\text{Left}(a,b)\}.$$

In other words, $X^r(a,b)$ is the set of points in X on the right-hand side of (a,b), while $X^l(a,b)$ is the set of points in X on the left-hand side of (a,b).

Definition 1. *Given a set $X \subset \mathbb{R}^2$ of cardinality $n \geq 4$ in general position, we say that the oriented edge (a,b) with $a, b \in X$, $a \neq b$, is a $(k-1)$-*set edge *if $|X^r(a,b)| = k-1$ (and hence $|X^l(a,b)| = n-k-1$).*

Setting $A = X^r(a,b) \cup \{a\}$ and $\bar{A} = X^l(a,b) \cup \{b\}$, it is clear that A is a k-set of X and the straight line (a,b) is the unique bitangent between $\text{Conv}(A)$ and $\text{Conv}(\bar{A})$ that keeps $A \setminus \{a\}$ on its right. Indeed, as illustrated in Figure 1a, two of the other three bitangents between $\text{Conv}(A)$ and $\text{Conv}(\bar{A})$ do not separate A from $\bar{A}$, and the remaining one does not keep $A \setminus \{a\}$ on its right-hand side. This proves the following proposition for any set $X \subset \mathbb{R}^2$ of $n \geq 4$ points and any $k = 2, \ldots, \lfloor n/2 \rfloor$.

Proposition 3. *There exists a bijection between the $(k-1)$-set edges of X and the k-sets of X: each $(k-1)$-set edge (a,b) can be associated with the k-set $\{a\} \cup X^r(a,b)$, while any k-set A corresponds to the unique oriented edge (a,b), bitangent to $Conv(A)$ and $Conv(\bar{A})$, such that $A \setminus \{a\} = X^r(a,b)$.*

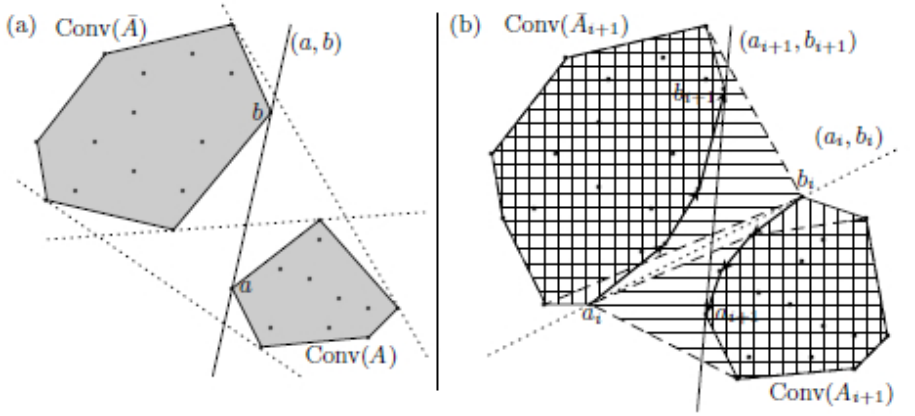

Fig. 1. (a) There are 4 bitangents between Conv(A) and Conv($\bar{A}$), but only (a, b) separates A and $\bar{A}$ keeping $A \setminus \{a\}$ on its right-hand side. (b) Given the k-set A_i (polygon with horizontal lines) associated to the $(k-1)$-set edge (a_i, b_i) we can compute the subsequent k-set A_{i+1} (polygon with vertical lines) associated to (a_{i+1}, b_{i+1}) by removing a_i and inserting b_i in the convex hull. Segments with arrow represent oriented edges scanned by procedure NextBitangent on the perimeter of the convex hull.

Let us denote by $A(a, b)$ the k-set corresponding to the $(k-1)$-set edge (a, b).

Since two $(k-1)$-set edges of X cannot have the same phase, we can consider the sequence

$$E_{k-1} = \{(a_1, b_1), ..., (a_h, b_h)\}$$

of all $(k-1)$-set edges of X ordered according with increasing phase.

The following proposition yields the key property for computing all k-sets $A(a_i, b_i)$ for $i = 1, 2, \ldots, h$. The proof is here omitted but its validity should be evident from Figure 1b.

Proposition 4. *For every $i = 1, 2, ..., h$, we have*

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

where $\langle i \rangle_h$ is the remainder of the division i/h.

As a consequence, setting

$$S_i = \sum_{p \in A(a_i, b_i)} p \tag{1}$$

we have $S_{1+\langle i \rangle_h} = S_i - a_i + b_i$.

Thus, in order to solve the 2-SCC-2 problem for an instance (X, k), we can simply determine the first oriented edge $(a_1, b_1) \in E_{k-1}$ and design a procedure for computing $(a_{1+\langle i \rangle_h}, b_{1+\langle i \rangle_h})$ from (a_i, b_i). This allows us to compute all pairs $(a_i, b_i) \in E_{k-1}$ and hence all values $\|S_i\|$'s one after the other, taking the largest one. The overall computation is described by Algorithm 2.

Let us start by showing the computation of (a_1, b_1).

Proposition 5. *Given a set $X \subset \mathbb{R}^2$ of $n \geq 4$ points in general position, for any $k = 2, \ldots, \lfloor n/2 \rfloor$ the $(k-1)$-set edge of smallest phase can be obtained in $O(n \log n)$ time.*

Proof. First, it is easy to determine the k-th smallest point a in X with respect to the total order $<_o$, together with set $A = \{q \in X \mid q \leq_o a\}$. Clearly, A is a k-set of X. Similarly, we can determine the $(k+1)$th element b in X with respect to $<_o$, i.e. the smallest point in the set $\bar{A} = X \setminus A$. To determine the $(k-1)$-set edge of A, the algorithm first computes the convex hulls $\mathrm{Conv}(A)$ and $\mathrm{Conv}(\bar{A})$. Then, starting from a and b, it moves two points u and v counter-clockwise on the perimeter of $\mathrm{Conv}(A)$ and $\mathrm{Conv}(\bar{A})$, respectively, stopping at the first edge on $\mathrm{Conv}(A)$ (respectively, on $\mathrm{Conv}(\bar{A})$) that is on the right (respectively, on the left) of the oriented edge (u, v).

The procedure is formally described by Algorithm 1 given below, where for every point p on the perimeter of a convex hull, $\mathrm{Succ}(p)$ is the counter-clockwise successor of p on the same perimeter.

Algorithm 1. FirstSetEdge(a,b)

```
 1: u := a; u' := Succ(u)
 2: v := b; v' := Succ(v)
 3: while ((u,u')Left(u,v) ∨ (v,v')Right(u,v)) do
 4:    if (u,u')Left(u,v) then
 5:       if (u,u')Left(u,v') then
 6:          u := u'; u' :=Succ(u)
 7:       else
 8:          v := v'; v' :=Succ(v)
 9:    else
10:       v := v'; v' :=Succ(v)
11: return (u,v)
```

At each loop iteration the procedure checks whether the current edges (u, u') and (v, v') on the two perimeters verify the exit condition

$$(u, u')\mathrm{Right}(u, v) \ \wedge \ (v, v')\mathrm{Left}(u, v),$$

which guarantees $A \setminus \{u\} = X^r(u, v)$ and $\bar{A} \setminus \{v\} = X^l(u, v)$. Hence, in the affirmative case, (u, v) is the required $(k-1)$-set edge.

The most expensive operation in the overall procedure is the computation of the convex hulls $\mathrm{Conv}(A)$ and $\mathrm{Conv}(\bar{A})$, which can be done in $O(n \log n)$ time [17]. Also note that the while loop at lines 3–10 requires $O(n)$ steps. □

Once the initial $(k-1)$-set edge is determined our general procedure computes all the subsequent $(k-1)$-set edges in the order defined by E_{k-1}. For each $(a_i, b_i) \in E_{k-1}$, the procedure computes the squared norm of S_i (defined in Equation 1), maintaining in q the largest value. The details are given in Algorithm 2, where procedure NextBitangent, called at lines 10 and 16, yields the successive $(k-1)$-set edge in the required order.

Algorithm 2. Solving 2-SCC-2

Input: a set $X \subset \mathbb{R}^2$ of $n \geq 4$ points in general position; an integer $1 < k \leq \lfloor n/2 \rfloor$.
Output: the solution $\pi = \{A, B\}$ of the 2-SCC-2 problem on instance X with constraint $|A| = k$.

1: Compute the k-th smallest point a in X with respect to $<_o$
2: $A := \{p \in X : p \leq_o a\}$; $\bar{A} := X \setminus A$
3: Compute the smallest point b in $\bar{A}$ with respect to $<_o$
4: $\mathcal{A} := \text{Conv}(A)$
5: $\bar{\mathcal{A}} := \text{Conv}(\bar{A})$
6: $(a_1, b_1) := \text{FirstSetEdge}(a, b)$
7: $S := \sum_{x \in A} x$
8: $q := \|S\|^2$
9: $(x, y) := (a_1, b_1)$
10: $(r, s) := \text{NextBitangent}(a_1, b_1)$
11: **while** $(r, s) \neq (a_1, b_1)$ **do**
12: $\quad S := S - r + s$
13: $\quad$ **if** $q < \|S\|^2$ **then**
14: $\quad\quad q := \|S\|^2$
15: $\quad\quad (x, y) := (r, s)$
16: $\quad (r, s) := \text{NextBitangent}(r, s)$
17: $\pi := \{X^r(x, y) \cup \{x\}, X^l(x, y) \cup \{y\}\}$
18: **return** π

Procedure NextBitangent is defined by Algorithm 3, which uses function *Succ* as in Algorithm 1. Such a procedure first computes the convex hulls $\mathcal{A}$ and $\bar{\mathcal{A}}$ of the new k-set A and of its complement by two insert and delete operations. Then, in the main loop, the procedure determines the $(k-1)$-set edge of the new k-set by following a path counter-clockwise on the perimeter of the two convex hulls. As in Algorithm 1, the exit condition

$$(u, u')\text{Right}(u, v) \ \wedge \ (v, v')\text{Left}(u, v)$$

guarantees that (u, v) is the required $(k-1)$-set edge.

The correctness proof of Algorithm 2, which is here omitted for lack of space, is based on the fact that distinct k-sets must have $(k-1)$-set edges with different phases.

The analysis of time complexity of Algorithm 2 requires the following result on the number of k-sets of X, given in [7].

Theorem 1. *For any set X of n points in $\mathbb{R}^2$ and any $k \in \mathbb{N}$, $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$, the number of k-sets of X is less than $6.48n\sqrt[3]{k}$.*

Proposition 6. *The time complexity required by Algorithm 2 on an input of $n \geq 4$ points with $1 < k \leq \lfloor \frac{n}{2} \rfloor$ is $O(n\sqrt[3]{k} \cdot \log^2 n)$.*

Proof. First recall that, by Proposition 5, the first part of the procedure from line 1 to line 8, can be executed in time $O(n \log n)$. The remaining time required

Algorithm 3. NextBitangent(r, s)

Input: a $(k-1)$-set edge (r, s) of X computed in Algorithm 2.
Output: the subsequent $(k-1)$-set edge of X in phase order.

```
 1: A := Insert(Delete(A, r),s)
 2: Ā := Insert(Delete(Ā, s),r)
 3: u := s; u' := Succ(u)
 4: v := r; v' := Succ(v)
 5: repeat
 6:    if (u, u')Left(u, v) then
 7:       if (u, u')Left(u, v') then
 8:          u := u'; u' :=Succ(u)
 9:       else
10:          v := v'; v' :=Succ(v)
11:    else
12:       v := v'; v' :=Succ(v)
13: until (u, u')Right(u, v) ∧ (v, v')Left(u, v)
14: return (u, v)
```

by the algorithm is dominated by calls to procedure NextBitangent. The computation maintains, as permanent structure, the convex hulls $\mathcal{A}$ and $\bar{\mathcal{A}}$, which are updated at lines 1–2 of each call of NextBitangent in $O(\log^2 n)$ time by using the data structure introduced in [16]. Since there is just one call to NextBitangent for each $(k-1)$-set edge, by Proposition 3 and Theorem 1 the total cost of all updates of $\mathcal{A}$ and $\bar{\mathcal{A}}$ is $O(n\sqrt[3]{k}\log^2 n)$.

The time cost of the other operations of NextBitangent is due to the repeat-until loop of Algorithm 3. Here, the key observation is that each edge (u, u') inside the repeat-until loop is a $(n-k)$-set edge, because the phase of the opposite edge (u', u) is included between the phases of two consecutive $(k-1)$-set edges. These edges (u, u') scan counter-clockwise the boundary of $\mathcal{A}$ and each of them is considered just by one call of NextBitangent. The same occurs for the k-set edges (v, v') scanning counter-clockwise the boundary of $\bar{\mathcal{A}}$. Therefore, the time required by the main loop in all calls to NextBitangent is at most proportional to $|E_k| + |E_{n-k}| = 2|E_k|$, which is again $O(n\sqrt[3]{k})$ by Theorem 1. Thus, the time cost of all calls to NextBitangent turns out to be $O(n\sqrt[3]{k}\log^2 n)$. □

5 Algorithm for the Full Problem

In this section we present an algorithm that, for an input $X \subset \mathbb{R}^2$ of n points in general position, computes an optimal 2-clustering $\{A_k, \bar{A}_k\}$ of X such that $|A_k| = k$, for each $k = 1, 2, \ldots, \lfloor n/2 \rfloor$. We prove that the algorithm works in time $O(n^2 \log n)$.

The result is based on Propositions 1 and 2 and on the following relationship between oriented edges and 2-clusterings of X including a k-set. Given two

distinct points $a, b \in X$, we associate the oriented edge (a, b) with the 2-clustering $\{A, \bar{A}\}$ of X where either A or $\bar{A}$ equals the set $\mathcal{R}(a, b)$ defined by

$$\mathcal{R}(a,b) = \begin{cases} X^r(a,b) \cup \{b\} & \text{if } a <_o b \\ X^l(a,b) \cup \{b\} & \text{otherwise.} \end{cases}$$

Note that here $\{A, \bar{A}\}$ is an unordered pair of sets. Moreover, A and $\bar{A}$ are, respectively, a k-set and a $(n-k)$-set for some $k \in \{1, 2, \ldots, n-1\}$. Also observe that the 2-clusterings associated with (a, b) and (b, a) are always different.

Proposition 7. *Given a set $X \subset \mathbb{R}^2$ of n points in general position, let $\{A, \bar{A}\}$ be a 2-clustering of X where A is a k-set for some $k \in \{1, 2, \ldots, n-1\}$. Then $\{A, \bar{A}\}$ is the 2-clustering of X associated with an oriented edge (a, b) with $a, b \in X$.*

Proof. Since A is a k-set we can consider a bitangent (u, v) that separates A and $\bar{A}$ with $u \in \bar{A}$ and $v \in A$. Assume that $u <_o v$: if $A = X^r(u, v) \cup \{v\}$ then (u, v) is the required oriented edge because $A = \mathcal{R}(u, v)$; otherwise, A equals $X^l(u, v) \cup \{v\}$ and the same 2-clustering $\{A, \bar{A}\}$ is associated with (v, u) because $\bar{A} = \mathcal{R}(v, u)$. A symmetric reasoning holds in case $v <_o u$. □

Note that in the previous proof we can choose the bitangent (u, v) separating A and $\bar{A}$ in two different ways. This proves that every k-sets of X is associated with two oriented edges. Hence, such a correspondence is quite different from the bijection of Proposition 3 introduced in the previous section.

By the proposition above, one can design an algorithm that scans all k-sets by considering in some order all oriented edge outgoing from each point. In order to compute efficiently the weights of the clusters we introduce a special order among the oriented edges $E = \{(a, b) \mid a, b \in X, a \neq b\}$: for every $(u, v), (w, z) \in E$, we set $(u, v) <_e (w, z)$ if either $u <_o w$ or $u = w$ and $slope(u, v) < slope(w, z)$.

Proposition 8. *For any instance set $X \subset \mathbb{R}^2$ of n points, the 2-SCC-2 problem for all $k = 1, 2, \ldots, \lfloor n/2 \rfloor$ can be solved in $O(n^2 \log n)$ time.*

Proof. Consider procedure Full 2-SCC-2 defined by Algorithm 4. It computes points $S_{\mathcal{R}(a,b)}$ (as defined in Proposition 1) for all oriented edges in $(a, b) \in E$, taken according with the total order $<_e$. For each $k = 1, 2, \ldots, \lfloor n/2 \rfloor$, the procedure maintains the optimal value $q[k] = \|S_A\|^2/k(n-k)$, where A is a k-set of X and keeps in $e[k]$ the associated oriented edge. By Propositions 2 and 7, this guarantees that all possible 2-clusterings of X are considered.

The computation first considers all points in X in the order $<_o$ and, for each $a \in X$, it determines $R = S_{A(a)}$, where $A(a) = \{u \in X \mid u <_o a\}$. Then, it computes $S_{\mathcal{R}(a,b)}$ for every edge (a, b) such that $b \in X \setminus \{a\}$ in the order $<_e$: for any pair of consecutive edges (a, b), (a, c), the value $S_{\mathcal{R}(a,c)}$ is obtained from $S_{\mathcal{R}(a,b)}$ by adding or subtracting c according whether $a <_o c$ or $c <_o a$ (see instructions 10 and 14, respectively). Note that such a computation only requires constant time.

The time complexity of the algorithm is dominated by the operation of sorting the oriented edges with the same starting point a. This can be done in $O(n \log n)$ time for each $a \in X$. Note that the other operations, in the inner for-loop, require at most constant time and hence they are executed $O(n^2)$ many times. Therefore, the overall time of the algorithm is $O(n^2 \log n)$. □

Algorithm 4. Full 2-SCC-2

Input: a set $X \subset \mathbb{R}^2$ of n points in general position
Output: the sequence $(e[1], \ldots, e[\lfloor n/2 \rfloor])$ of oriented edges, where each $e[k]$ is associated with the solution $\{A_k, \bar{A}_k\}$ of 2-SCC-2 for X such that $|A_k| = k$

1: **for** $k = 1, 2, \ldots, \lfloor n/2 \rfloor$ **do**
2: $\quad q[k] := 0$
3: $T := \sum_{p \in X} p$
4: Sort X according with $<_o$ and let $(a_1, a_2, \ldots, a_n)$ be the ordered sequence
5: **for** $i = 1, 2, \ldots, n$ **do**
6: $\quad R := \sum_{j<i} a_j$
7: $\quad g := i - 1$
8: $\quad$ Sort the set $X \setminus \{a_i\}$ according with $slope(a_i, \cdot)$ and let $(b_1, b_2, \ldots, b_{n-1})$ be the ordered sequence
9: $\quad$ **for** $j = 1, 2, \ldots, n-1$ **do**
10: $\quad\quad$ **if** $a_i <_o b_j$ **then** $\begin{cases} R := R + b_j \\ g := g + 1 \end{cases}$
11: $\quad\quad$ **if** $g \leq n - g$ **then** $\begin{cases} m := g \\ S := R \end{cases}$
12: $\quad\quad$ **else** $\begin{cases} m := n - g \\ S := T - R \end{cases}$
13: $\quad\quad$ **if** $q[m] < \frac{\|S\|^2}{g(n-g)}$ **then** $\begin{cases} q[m] := \frac{\|S\|^2}{g(n-g)} \\ e[m] := (a_i, b_j) \end{cases}$
14: $\quad\quad$ **if** $b_j <_o a_i$ **then** $\begin{cases} R := R - b_j \\ g := g - 1 \end{cases}$
15: **return** $(e[1], \ldots, e[\lfloor n/2 \rfloor])$

References

1. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of Euclidean sum-of-squares clustering. Machine Learning 75, 245–249 (2009)
2. Basu, S., Davidson, I., Wagstaff, K.: Constrained Clustering: Advances in Algorithms, Theory, and Applications. Chapman and Hall/CRC (2008)
3. Bertoni, A., Goldwurm, M., Lin, J., Pini, L.: Size-constrained 2-Clustering in the Plane with Manhattan Distance. In: Proc. 15th Italian Conference on Theoretical Computer Science. CEUR Workshop Proceedings, vol. 1231, pp. 33–44. CEUR-WS.org (2014) ISSN 1613-0073
4. Bertoni, A., Goldwurm, M., Lin, J., Saccà, F.: Size Constrained Distance Clustering: Separation Properties and Some Complexity Results. Fundamenta Informaticae 115(1), 125–139 (2012)

5. Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)
6. Dasgupta, S.: The hardness of k-means clustering. Technical Report CS2007-0890, Department of Computer Science and Engineering, University of California, San Diego (2007)
7. Dey, T.: Improved Bounds for Planar k-Sets and Related Problems. Discrete & Computational Geometry 19(3), 373–382 (1998)
8. Edelsbrunner, H.: Algorithms in Combinatorial Geometry. EATCS monographs on theoretical computer science. Springer (1987)
9. Erdős, P., Lovász, L., Simmons, A., Straus, E.G.: Dissection graphs of planar point sets. In: A Survey of Combinatorial Theory (Proc. Internat. Sympos., Colorado State Univ., Fort Collins, Colo., 1971), pp. 139–149. North-Holland, Amsterdam (1973)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer (2009)
11. Inaba, M., Katoh, N., Imai, H.: Applications of weighted voronoi diagrams and randomization to variance-based k-clustering (extended abstract). In: Proceedings of the Tenth Annual Symposium on Computational Geometry, SCG 1994, USA, pp. 332–339 (1994)
12. Lin. J.: Exact algorithms for size constrained clustering. PhD Thesis, Dottorato di ricerca in Matematica, Statistica e Scienze computationali, Università degli Studi di Milano. Ledizioni Publishing (2013)
13. MacQueen, J.B.: Some method for the classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Structures, pp. 281–297 (1967)
14. Mahajan, M., Nimbhorkar, P., Varadarajan, K.: The planar k-means problem is NP-hard. Theoretical Computer Science 442, 13–21 (2012)
15. Novick, B.: Norm statistics and the complexity of clustering problems. Discrete Applied Mathematics 157, 1831–1839 (2009)
16. Overmars, M.H., van Leeuwen, J.: Maintenance of configurations in the plane. J. Comput. Syst. Sci. 23(2), 166–204 (1981)
17. Preparata, F., Shamos, M.: Computational geometry: an introduction. Texts and monographs in computer science. Springer (1985)
18. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic Press, Elsevier (2009)
19. Vattani, A.: K-means requires exponentially many iterations even in the plane. In: Proceedings of the 25th Symposium on Computational Geometry (SoCG) (2009)
20. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proc. of the 17th Intl. Conf. on Machine Learning, pp. 1103–1110 (2000)
21. Zhu, S., Wang, D., Li, T.: Data clustering with size constraints. Knowledge-Based Systems 23(8), 883–889 (2010)