Efficient Algorithms for Detecting Regular Point Configurations

Luzi Anderegg[1], Mark Cieliebak[1], and Giuseppe Prencipe[2]

[1] ETH Zurich, {anderegg,cieliebak}@inf.ethz.ch
[2] Università di Pisa, prencipe@di.unipi.it (contact author)

Abstract. A set of n points in the plane is in *equiangular configuration* if there exist a center and an ordering of the points such that the angle of each two adjacent points w.r.t. the center is $\frac{360^\circ}{n}$, i.e., if all angles between adjacent points are equal. We show that there is at most one center of equiangularity, and we give a linear time algorithm that decides whether a given point set is in equiangular configuration, and if so, the algorithm outputs the center. A generalization of equiangularity is *σ-angularity*, where we are given a string σ of n angles and we ask for a center such that the sequence of angles between adjacent points is σ. We show that σ-angular configurations can be detected in time $O(n^4 \log n)$.

Keywords: Weber point, equiangularity, σ-angularity, design of algorithms, computational geometry.

1 Introduction

We study how to identify geometric configurations that are in a sense generalizations of stars: a set of n distinct points P in the plane is in *equiangular configuration* if there exists a point $c \notin P$ — the *center of equiangularity* — and an ordering of the points such that each two adjacent points form an angle of $\frac{360^\circ}{n}$ w.r.t. c (see Figure 1(a)).

Obviously, if all points have the same distance from the center, then they form a regular star. Note that we exclude the special case that any of the given points is at center c. Furthermore, observe that the number of points in equiangular configurations can be odd or even, and that any set of two points is always in equiangular configuration. In the remainder of this paper, we will consider only point sets with at least three points.

There is a strong connection between equiangular configurations and *Weber points*, which are defined as follows: a point w is a Weber point of point set P if it minimizes $\sum_{p \in P} |p - x|$ over all points x in the plane, where $|p - x|$ denotes the Euclidean distance between p and x [8]. Hence, a Weber point minimizes the sum of all distances between itself and all points in P. The Weber point for an arbitrary point set is unique except for the case of an even number of points which are all on a line [3]. We will show that the center of equiangularity, if it exists, is a Weber point; thus, there is at most one center of equiangularity for

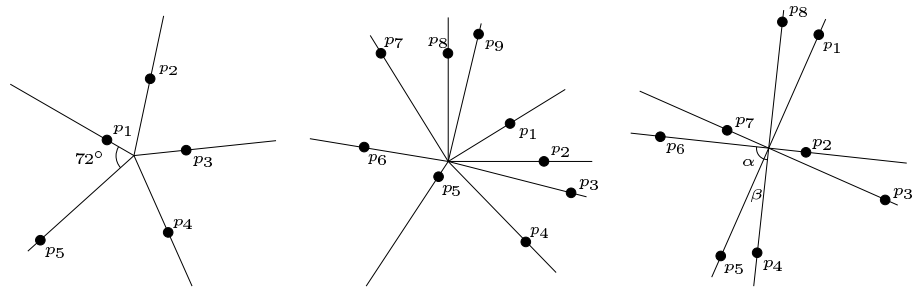

Fig. 1: Example of (a) equiangular configuration with $n = 8$; (b) σ-angular configuration with $n = 9$, where $\sigma = (31°, 45°, 14°, 31°, 49.5°, 68.5°, 76°, 31°, 14°)$; and (c) biangular configuration with $n = 8$.

$n \geq 3$ points that are not on a line. Obviously, we could check easily whether a given set of points is in equiangular configuration if we could find their Weber point. Unfortunately, no efficient algorithms are known to find the Weber point in general; even worse, it can be shown that the Weber point cannot even be computed using radicals [2]. Hence, other algorithms are necessary, which we will develop throughout this paper. The algorithm we will present for identifying equiangularity has an interesting implication on Weber points: If n points are in equiangular configuration, then we can use our algorithm to compute their Weber point. This is rather surprising, since such results are known for only few other patterns (e.g. all points are on a line), whereas this does not hold in general for many easy–looking geometric pattern, such as circles [5].

A generalization of equiangularity is *σ-angularity*, where we are given a string $\sigma = (\sigma_1, \ldots, \sigma_n)$ of n angles and we ask whether there exists a center c and an ordering of the points such that the sequence of angles between each two adjacent points w.r.t. the center is σ (see Figure 1(c)). Observe that the center of σ-angularity is not necessary unique. Obviously, equiangularity is equivalent to σ-angularity with $\sigma = (\frac{360°}{n}, \frac{360°}{n}, \ldots, \frac{360°}{n})$. The case of two alternating angles α and β, i.e., $\sigma = (\alpha, \beta, \alpha, \ldots, \beta)$, is referred to as *biangular* (see Figure 1(b)).

σ-angular configurations have been applied successfully in robotics, namely in solving the Gathering Problem, which – informally – can be defined as follows: given is a set of autonomous mobile robots that cannot communicate at all and that can only observe the positions of all other robots in the plane. The task is to gather the robots at an arbitrary point in the plane that is not fixed in advance. One of the main difficulties of this problem is to deal with configurations that are totally symmetric, for instance where the robots' positions form a regular[3] n-gon. Recently, an algorithm solving the Gathering Problem has been proposed which uses – among other techniques – the center

[3] Note that a regular n-gon is a special case of equiangular configuration.

of equiangularity or biangularity to gather the robots there [4]. Hence, efficient algorithms are needed to find the centers of such configurations.

In this paper, we present algorithms that decide whether a point set is in σ-angular configuration, for a given string σ, and if so, the algorithms output a corresponding center. To our knowledge, there is no treatment of equiangularity in the vast amount of literature on computational geometry. Only very distantly related, if at all, are for instance star-shaped polygons and star–graphs [7].

For the general case of σ-angularity, we will present in Section 2 an algorithm with running time $O(n^4 \log n)$. For the special cases of biangular and equiangular configurations, this algorithm runs in cubic time, if the two angles are given. In Section 3, we will give another algorithm that allows to detect equiangular configurations even in linear time. All algorithms are straightforward to implement.

2 σ-Angular Configurations

In this section, we present an algorithm that detects σ-angularity in running time $O(n^4 \log n)$, and we show how to simplify this algorithm for biangular and equiangular configurations, yielding running time $O(n^3)$. Our algorithms rely on the notion of *Thales circles*, which we introduce below.

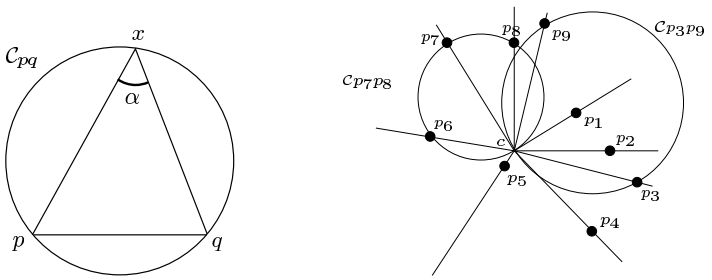

Fig. 2: (a) A Thales circle of angle $\alpha < 90°$. (b) $\mathcal{C}_{p_3p_9}$ is Thales circle for p_3 and p_9 with $\alpha = \sigma_9 + \sigma_1 + \sigma_2$, $\mathcal{C}_{p_7p_8}$ is Thales circle for p_7 and p_8 with $\alpha = \sigma_7$.

2.1 Thales Circles

Given two points p and q and an angle $0° < \alpha < 180°$, a circle $\mathcal{C}$ is a *Thales circle of angle α for p and q* if p and q are on $\mathcal{C}$, and there is a point x on $\mathcal{C}$ such that $\sphericalangle(p, x, q) = \alpha$, where $\sphericalangle(p, x, q)$ denotes the angle between p and q w.r.t. x. In the following we will denote such a circle also by $\mathcal{C}_{pq}$. An example of a Thales circle[4]

[4] The name "Thales circle" refers to Thales of Miletus, who was one of the first to show that all angles in a semi-circle have 90°.

can be found in Figure 2(a). It is well-known from basic geometry that all angles on a circle arc are identical, i.e., given a Thales circle $\mathcal{C}$ of angle α for points p and q such that $\sphericalangle(p,x,q) = \alpha$ for some point x on $\mathcal{C}$, then $\sphericalangle(p,x',q) = \alpha$ for every point x' on the same circle arc of $\mathcal{C}$ where x is (cf. Figure 2(a)).

Lemma 1. *Given two points p and q and an angle $0° < \alpha < 180°$, the Thales circles of angle α for p and q can be constructed in constant time.*

The lemma can be proven using basic geometry.

Observe that for two points p and q, there is exactly one Thales circle of angle 90°, and there are two Thales circles of angle $\alpha \neq 90°$. However, in the remainder of this paper we will often speak of "*the* Thales circle", since it will be clear from the context which of the two Thales circles we refer to.

The connection between Thales circles and σ-angular configurations is the following (see Figure 2(b)): Assume that a point set P is in σ-angular configuration with center c, for some string of angles $\sigma = (\sigma_1, \ldots, \sigma_n)$. For two points $p, q \in P$, let α be the angle between p and q w.r.t. c, i.e., $\alpha = \sphericalangle(p,c,q)$. Then $\alpha = \sum_{k=i}^{j} \sigma_k$ for two appropriate indices i, j (taken modulo n). Let $\mathcal{C}$ be the circle through p, q and c. Then $\mathcal{C}$ is a Thales circle of angle α for p and q. Since this holds for any pair of points from P - with appropriate angles - this yields the following:

Observation 1 *The center of σ-angularity must lie in the intersection of Thales circles for any two points from P of appropriate angle.*

We will use this observation in our algorithms to find candidates for the center of σ-angularity.

2.2 Algorithm for σ-Angular Configurations

We now present an algorithm that detects σ-angular configurations in time $O(n^4 \log n)$; before, we observe basic properties of σ-angular configurations that we apply in our algorithm.

Let σ be a string of angles and P be a point set that is in σ-angular configuration with center c. First, observe that the angles in σ must sum up to 360°; thus, there can be at most one angle in σ that is larger than 180°. Let α_{max} be the maximum angle in σ. Then the following holds (cf. Figure 2.2): If $\alpha_{max} > 180°$, then center c is outside the convex hull of P; if $\alpha_{max} = 180°$ and there is no other angle of 180° in σ, then center c is on the convex hull; if there are two angles of 180° in σ, then all other angles in σ are of 0°, and the points in P are on a line; and, finally, if $\alpha_{max} < 180°$, then center c is strictly inside the convex hull of P. Furthermore, observe for the last case – where c is inside the convex hull – that the ordering of the points in P that yields σ, if restricted to the points on the convex hull of P, corresponds to the ordering of these points along the convex hull.

Our algorithm to detect σ-angularity will distinguish these four cases. For each case, the algorithm will generate few candidate points that might be a

center of σ-angularity. Then we test for each candidate c whether it is indeed a center as follows: We compute the angle between a fixed point on the convex hull, say x, and every other point $p \in P$ w.r.t. c. We sort these angles and compute sequence τ by subtracting each angle from its successor. Then candidate c is a center of σ-angularity if sequence τ or its reverse is a cyclic shift of σ. This test requires time $O(n \log n)$.

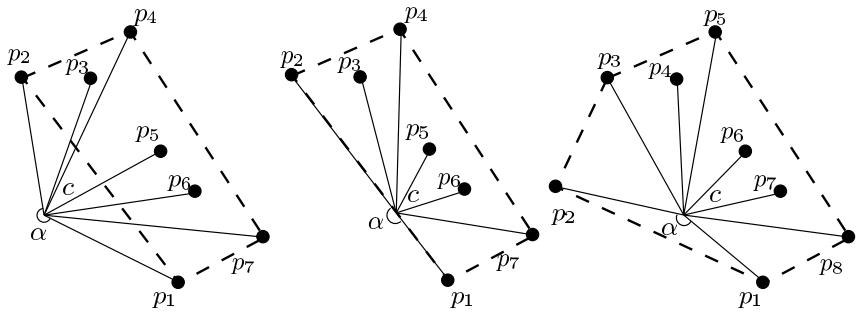

Fig. 3: σ-angular configurations with (a) $\alpha_{max} > 180°$, (b) α_{max} unique angle of 180°, and (c) $\alpha_{max} < 180°$ (dashed lines show the convex hull of $\{p_1, \ldots, p_n\}$).

Theorem 1. *Given a point set P of $n \geq 3$ distinct points in the plane and a string $\sigma = (\sigma_1, \ldots, \sigma_n)$ of n angles with $\sum_i \sigma_i = 360°$, there is an algorithm with running time $O(n^4 \log n)$ that decides whether the points are in σ-angular configuration, and if so, the algorithm outputs a center of σ-angularity.*

Proof. The algorithm consists of different routines, depending on whether the largest angle in σ is greater than, equal to, or less than 180°. We present the algorithm for the case that all angles are less than 180°. The other cases are solved similarly.

Case All angles less than 180°. In this case, a center of σ-angularity, if it exists, is strictly inside the convex hull of P (see Figure 2.2(c)). Moreover, if we fix three points from P and compute the Thales circles for these points with appropriate angles, then the center of equiangularity lies in the intersection of these circles (cf. Observation 1). This idea is implemented in Algorithm 1.

To see the correctness of the algorithm, note that α and β are two consecutive range sums of σ' that sum over at most n angles. Thus, the algorithm has a loop for each angle α that might occur between x and y according to σ.

If for some angle α all points from P are on the Thales circle $\mathcal{C}_{xy}$, then α cannot be the angle between x and y, since the center of σ-angularity would have to be both on circle $\mathcal{C}_{xy}$ and inside the convex hull of the points, which is not possible (recall that the center cannot be a point from P by definition).

Algorithm 1 Algorithm for all angles less than 180°.

If the points in P are on a line **Then**
  Return ≪not σ-angular≫
$\sigma' = (\sigma_1, \ldots, \sigma_n, \sigma_1, \ldots, \sigma_n)$
x, y = two arbitrary neighbors on convex hull of P
For $i = 1$ **To** n **Do**
  For $j = i$ **To** $i + n - 1$ **Do**
    $\alpha = \sum_{l=i}^{j} \sigma'_l$
    $\mathcal{C}_{xy}$ = Thales circle of angle α for x and y
    If at least one point from P is not on $\mathcal{C}_{xy}$ **Then**
      z = arbitrary point from P that is not on $\mathcal{C}_{xy}$
      For $k = j + 1$ TO $j + n$ **Do**
        $\beta = \sum_{l=j+1}^{k} \sigma'_l$
        $\mathcal{C}_{xz}$ = Thales circle of angle β for x and z
        If $\mathcal{C}_{xy}$ and $\mathcal{C}_{xz}$ intersect in two points **Then**
          c = point in intersection that is not x
          If c is a center of σ-angularity **Then**
            Return ≪σ-angular with center c≫
        $\mathcal{C}_{yz}$ = Thales circle of angle β for y and z
        If $\mathcal{C}_{xy}$ and $\mathcal{C}_{yz}$ intersect in two points **Then**
          c = point in intersection that is not y
          If c is a center of σ-angularity **Then**
            Return ≪σ-angular with center c≫
Return ≪not σ-angular≫

The running time of Algorithm 1 is $O(n^4 \log n)$. □

We now show how to simplify the previous algorithm to test in only cubic time whether a point set is in biangular configuration, presumed we know the two corresponding angles:

Corollary 1. *For two given angles α and β with $0° \leq \alpha, \beta \leq 180°$, biangular configurations can be detected in time $O(n^3)$.*

Proof. First observe that the special case where either α or β is of 180° is easy to solve: In this case, the other angle has to be 0°, the point set P consists of exactly 4 points, all points are on a line, and the center of biangularity is every point between the two median points in P.

In the following, we assume that $\alpha, \beta < 180°$, and adapt Algorithm 1 from above for this special case: First, we pick two neighbor points x and y on the convex hull of P. If the points in P are in biangular configuration, then the angle between x and y is $\gamma = k \cdot (\alpha + \beta) + \delta$, for some value $k \in \{0, \ldots, \frac{n}{2}\}$ and some angle $\delta \in \{0°, \alpha, \beta\}$. For each of these possibilities, we compute the corresponding Thales circle $\mathcal{C}_{xy}$ of angle γ. Like in Algorithm 1, if for some k and δ all other points are on $\mathcal{C}_{xy}$, then these values cannot be the right choice. Otherwise, we pick a point z that is not on $\mathcal{C}_{xy}$. The angle between y and z is $\gamma' = k' \cdot (\alpha + \beta) + \delta'$, for appropriate values $k' \in \{0, \ldots, \frac{n}{2}\}$ and $\delta' \in \{0°, \alpha, \beta\}$.

We then compute all corresponding Thales circles $\mathcal{C}_{yz}$ of angle γ' and check for each of the (at most) two points in the intersection between C_{xy} and C_{yz} whether it is a center of biangularity. The correctness follows from Observation 1.

The running time of this algorithm is $O(n^3)$, instead of $O(n^4 \log n)$ for Algorithm 1. This improvement results from two facts: First, instead of considering all combinations of two consecutive range sums of σ in Algorithm 1, we consider only all combinations of $k(\alpha+\beta)+\delta$ and $k'(\alpha+\beta)+\delta'$, where $k, k' \in \{1, \ldots, \frac{n}{2}\}$ and $\delta, \delta' \in \{0°, \alpha, \beta\}$. This reduces the number of executions of the inner loop from $O(n^3)$ to $O(n^2)$.

Second, within the loop, we can test a candidate c in linear time (instead of $O(n \log n)$) as follows: we compute all angles between x and p w.r.t. c for all points $p \in P$. If any of these angles is not of the form $k(\alpha+\beta)+\delta$ with $k \in \{0, \ldots, \frac{n}{2}\}$ and $\delta \in \{0°, \alpha, \beta\}$, then c cannot be a center of biangularity. Otherwise, we use three arrays of length $\frac{n}{2}$ to store the points from p, each corresponding to one possible value of δ. More precisely, if the angle between x and p is $k \cdot (\alpha+\beta)+\delta$, then we store point p in position k of the corresponding array. This process resembles a kind of bucket counting sort (see e.g. [6]). The points are in biangular configuration with center c if and only if the following three conditions hold: 1. in the array corresponding to $\delta = 0°$ there is exactly one entry in each position; 2. one of the other two arrays is empty; and 3. in the remaining array there is exactly one entry in each position. □

Observe that if the configuration is biangular, then we can immediately determine the corresponding ordering of the points from the values in the three arrays from the algorithm above. Furthermore, note that the previous corollary yields immediately an algorithm for equiangularity with running time $O(n^3)$, since in this case $\alpha = \beta = \frac{360°}{n}$. In the next section we will give an algorithm that identifies equiangular configurations even in linear time.

3 Equiangular Configurations

In this section, we first show that there is at most one center of equiangularity. Then we give a linear time algorithm that detects equiangular configurations.

Lemma 2. *Given a point set P of $n \geq 3$ distinct points in the plane that are in equiangular configuration with center c, the following holds:*

1. *Center c is invariant under straight movement of any of the points towards to or away from c, i.e., the points remain in equiangular configuration with center c.*
2. *Center c is the Weber point of P.*
3. *The center of equiangularity is unique.*

Proof. The first claim is trivial. For the second claim, assume that we move the points in P straight in the direction of c until they occupy the vertices of regular n-gon centered in c (cf. Figure 3(a)). These "shifted" points are rotational

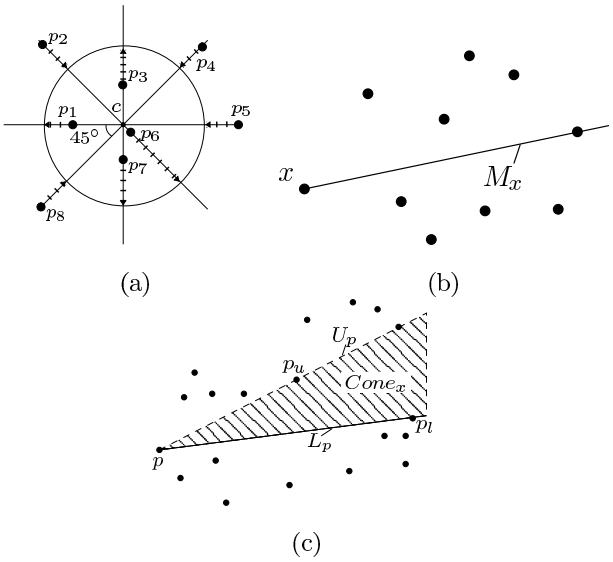

Fig. 4: (a) Equiangular configuration and its "shifted" points. (b) Median line M_x. (c) Median cone $Cone_x$.

symmetric, with symmetry center c. It is straightforward to see that the Weber point for these shifted points is c, since it must be at the center of rotational symmetry. Since the movements were in the direction of c, and the Weber point is invariant under straight movement of any of the points in its direction [1], c must be the Weber point of P. The third claim of the lemma follows immediately from the previous one and from the fact that the Weber point is unique if the points are not collinear [3] (observe that more than two points in equiangular configuration cannot be collinear). □

A similar argument can be used to show that the center of biangularity, if it exists, is unique. We now present an algorithm that identifies equiangular configurations in linear time.

Theorem 2. *Given a point set P of $n \geq 3$ distinct points in the plane, there is an algorithm with running time $O(n)$ that decides whether the points are in equiangular configuration, and if so, the algorithm outputs the center of equiangularity.*

Proof. We handle separately the cases of n even and n odd, where the first case turns out to be much easier. Both algorithms can be divided into two steps: First, they compute (at most) one candidate point for the center of equiangularity; then they check whether this candidate is indeed the center of equiangularity.

1. Case: n is even. The main idea for n even is as follows: assume for a moment that the points in P are in equiangular configuration with center c. Since n is

even, for every point $p \in P$ there is exactly one corresponding point $p' \in P$ on the line from p through c (there cannot be more than one point on this line, since this would imply an angle of $0°$ between these points and p w.r.t. c). Hence, the line through p and p' divides the set of points into two subsets of equal cardinality $\frac{n}{2} - 1$. We will refer to such a line as *median line* (cf. Figure 3(b)). Obviously, center of equiangularity c must lie in the intersection of the median lines of all points in P. We will use this observation in the following algorithm by first computing two median lines and then checking whether their intersection is the center of equiangularity.

Algorithm 2 Algorithm for n even.

1: $x =$ arbitrary point on the convex hull of P
2: $M_x =$ median line of x
3: **If** $|M_x \cap P| > 2$ **Then Return** $\ll$not equiangular$\gg$
4: $y =$ clockwise neighbor of x on the convex hull of P
5: $M_y =$ median line of y
6: **If** $|M_y \cap P| > 2$ **Then Return** $\ll$not equiangular$\gg$
7: **If** $M_x \cap M_y = \emptyset$ **Then Return** $\ll$not equiangular$\gg$
8: $c =$ unique point in $M_x \cap M_y$
9: **If** c is the center of equiangularity **Then Return** $\ll\sigma$-angular with center $c\gg$
10: **Else Return** $\ll$not equiangular$\gg$

Correctness. To see the correctness of Algorithm 2, observe the following:

For a point on the convex hull of P, the median line is unique. (This does not necessarily hold for inner points.) If there are more than two points from P on M_x, then the configuration cannot be equiangular. To see this, recall that the center of equiangularity, if it exists, will lie on M_x. If there are at least three points from P on M_x, then two of them will have angle $0°$ w.r.t. the center, which is impossible for equiangular configurations.

If the two median lines M_x and M_y are equal, then the configuration cannot be equiangular. To see this, first observe that in this case x and y are the only points from P on M_x. Since x and y are adjacent points on the convex hull of P, all points from P are on one side of M_x. On the other hand, since M_x is a median line, by definition the number of points from P on both sides of M_x is equal. Hence, $n = 2$, in contradiction to the assumption of the theorem.

On the other hand, if the two median lines M_x and M_y do not intersect, then the points are not in equiangular configuration. This is obvious, since the center of equiangularity will lie in the intersection of the median lines of all points from P.

Finally, if the points are in equiangular configuration with center c, then c must lie in the intersection of M_x and M_y.

Time Complexity. We now show how to implement each step of Algorithm 2 in linear time.

In order to find point x on the convex hull of P, we could just compute the entire convex hull, but this would take time $\Theta(n \log n)$. Instead, we take as x an arbitrary point from P with a minimal x-coordinate. This is always a point on the convex hull of P and can be found in linear time.

We can find M_x in linear time as follows. First, we compute the slopes of all lines from x to any other point $p \in P - \{x\}$ and store these slopes in an (unsorted) array. Then we pick the line with the median slope. This requires only linear time, since selecting the k-th element of an unsorted array - and consequently the median as well - can be done in linear time [6].

Using the unsorted array from the previous paragraph, we can choose $y \in P$ such that the line through x and y has maximal slope amongst all lines through x and a point in P, using the unsorted array from the previous step. If there is more than one candidate for y, then we take the one closest to x. Then y is a point on the convex hull of P.

Finally, the test whether c is the center of equiangularity can be done in linear time as follows, analogous to the test in the proof of Corollary 1: Let $\alpha = \frac{360^\circ}{n}$. We compute all angles between x and p w.r.t. c for all points $p \in P, p \neq x$. If any of these angles is not a multiple of α, then the points are not in equiangular configuration. Otherwise, we store the points from P in an array of length $n-1$, where we store point p in the array at position k if the angle between x and p is $k \cdot \alpha$. This process resembles again a kind of bucket counting sort (see e.g. [6]). If there is exactly one point in each position of the array, then c is the center of equiangularity; otherwise, the points in P are not in equiangular configuration.

2. Case: n is odd. For the odd case, we need to relax the concept of median line, and introduce that of *cone*.

The basic idea of the algorithm for this case is similar to the case "n is even", but slightly more sophisticated, since we have to relax the concept of median lines: assume for a moment that the points are in equiangular configuration with center c. For every point $p \in P$, there is no other point on the line from c to p, since n is odd and no angle of 0° can occur. Hence, such a line divides the set of points into two subsets of equal cardinality $\frac{n-1}{2}$. If we pick two points $p_l, p_u \in P$ that are "closest" to this line, in the sense that the slope of the line from p to c is between the slopes of lines L_p and U_p from p to p_l and p_u, respectively, then these two points define a cone with tip p (cf. Figure 3(c)). We will refer to this cone as *median cone*, since the number of points from P on each side of the cone (including lines L_p and U_p, respectively) equals $\frac{n-1}{2}$. Observe that the median cone is unique for points on the convex hull of P, and that the center of equiangularity, if it exists, lies in the intersection of all median cones of points on the convex hull of P. Moreover, for two points x and y on the convex hull of P, every point that is between x and y in the ordering of the points that yields equiangularity is a point in the area "between" $Cone_x$ and $Cone_y$ (bold points

Algorithm 3 Algorithm for n odd.

1: x = arbitrary point on the convex hull of P
2: $Cone_x$ = median cone with tip x
3: y = clockwise neighbor of x on the convex hull of P
4: $Cone_y$ = median cone of y
5: k_{xy} = number of points from $P - \{x, y\}$ that lie between the cones $Cone_x$ and $Cone_y$, including the boundaries
6: $\alpha = \frac{360°}{n}$
7: $\beta_{xy} = (k_{xy} + 1) \cdot \alpha$
8: $\mathcal{C}_{xy}$ = Thales circle for x and y of angle β_{xy}
9: Arc = circle arc $\mathcal{C}_{xy} \cap Cone_x \cap Cone_y$
10: g = line through the starting and ending point of Arc
11: H = halfplane defined by g that does not contain x and y
12: $S = H \cap P$
13: **If** $S = \emptyset$ **Then Return** ≪not equiangular≫
14: z = point from S with maximal perpendicular distance from g over all points in P
15: **If** $z \in \mathcal{C}_{xy}$ **Then Return** ≪not equiangular≫
16: k_{yz} = number of points from $P - \{x, y, z\}$ that lie between the cones $Cone_y$ and $Cone_z$, including the boundaries
17: $\beta_{yz} = (k_{yz} + 1) \cdot \alpha$
18: $\mathcal{C}_{yz}$ = Thales circle for y and z of angle β_{yz}
19: **If** $|\mathcal{C}_{xy} \cap \mathcal{C}_{yz}| = 1$ **Then Return** ≪not equiangular≫
20: c = unique point in $\mathcal{C}_{xy} \cap \mathcal{C}_{yz} - \{y\}$
21: **If** c is the center of equiangularity **Then Return** ≪σ-angular with center c≫
22: **Else Return** ≪not equiangular≫

in the upper left area in Figure 3(c)). We denote this number by k_{xy}. Notice that the angle between x and y w.r.t. c would be $(k_{xy} + 1) \cdot \alpha$.

The complete algorithm is shown in Algorithm 3, and illustrated in Figure 5. Its main idea is to elect three points x, y, z on the convex hull of P; to use the median cones of these points to determine the angels between the points w.r.t. the center of equiangularity (if it exists); to find one candidate center in the intersection of Thales circles for these points of appropriate angles; and, finally, to check whether this candidate is indeed the center of equiangularity.

Correctness. To see the correctness of Algorithm 3, observe the following:

By definition, no point from P can be strictly inside cones $Cone_x$ or $Cone_y$.

The center of equiangularity, if it exists, is inside the convex hull of the points. Thus, it is straightforward which of the two Thales circles for x and y is appropriate, since x and y are points on the convex hull. The same holds for the Thales circle for y and z.

After picking x and y, we need to find another point z on the convex hull such that the intersection of the two corresponding Thales circles yields a candidate for the center of equiangularity. For the choice of z we have to be careful: if we simply took z as the next point on the convex hull (after x and y), it might

happen that the corresponding Thales circle $\mathcal{C}_{yz}$ coincides with $\mathcal{C}_{xy}$, and, consequently, we do not obtain a single candidate for the center of equiangularity in their intersection. Therefore, we have to chose z such that it is on the convex hull and such that it does not lie on $\mathcal{C}_{xy}$. This is done in Lines 9–15, and discussed in the following (see Figure 5(b)).

If S is empty, then the points are not in equiangular configuration. To see this, observe that the center of equiangularity, if it exists, must be on Arc, with Arc as computed in Line 9. On the other hand, if $S = \emptyset$, then Arc is completely outside the convex hull of P, in contradiction to the fact that the center of equiangularity must be inside the convex hull.

Point z computed in Line 15 is on the convex hull of P: since, by construction, all points from P are on one side of the line through z that is parallel to g (line g' in Figure 5(b)).

If point z is on $\mathcal{C}_{xy}$, then the points in P are not in equiangular configuration. To see this, assume that $z \in \mathcal{C}_{xy}$. Then $z \in Arc$ by construction, hence, $z \in Cone_x \cap Cone_y$. By definition, no point from P can be strictly inside any median cone; thus, z has to be at one of the two endpoints of Arc. Moreover, no point from P can be strictly inside H, since H is delimited by g, and z is the point from P furthest away from g. This implies that points on Arc are on or outside the convex hull of P. On the other hand, only points on Arc might be a center of equiangularity, since such a center must lie in the intersection of $Cone_x, Cone_y$ and $\mathcal{C}_{xy}$. Since the center of equiangularity, if it exists, must be strictly inside the convex hull of P (otherwise there would be an angle of 180°), the points cannot be in equiangular configuration.

The two circles $\mathcal{C}_{xy}$ and $\mathcal{C}_{yz}$ are different, since $z \notin \mathcal{C}_{xy}$. Thus, they either intersect in one or two points. If $|\mathcal{C}_{xy} \cap \mathcal{C}_{yz}| = 1$, then y is the unique point in the intersection, and the points in P cannot be in equiangular configuration. On the other hand, if the center of equiangularity exists, then it is in $\mathcal{C}_{xy} \cap \mathcal{C}_{yz}$, and thus found in Line 20.

Time Complexity. We now show how to implement each step of Algorithm 2 in linear time.

Points x and y can be found in linear time like in the algorithm for the case "n is even".

To find the median cone $Cone_x$, we proceed analogous to the search for the median line in algorithm for the case "n is even": First we compute the slopes of all lines from x to every other point $p \in P$, and store them in an (unsorted) array. Let L_x and U_x be the two lines such that their slope is the $\lfloor n/2 \rfloor$-th and the $\lceil n/2 \rceil$-th largest, respectively, among all stored slopes. These two lines define $Cone_x$, and can be found in linear time using like before an algorithm to select the k-th element of an unsorted list [6]. Analogous, we can find $Cone_y$.

To determine value k_{xy}, let p be the point in the intersection of L_y and U_x, the two lines that delimit cones $Cone_x$ and $Cone_y$ (cf. Figure 5(a)). Then k_{xy} is the number of points from $P - \{x, y\}$ that lie inside or on the convex angle in p with

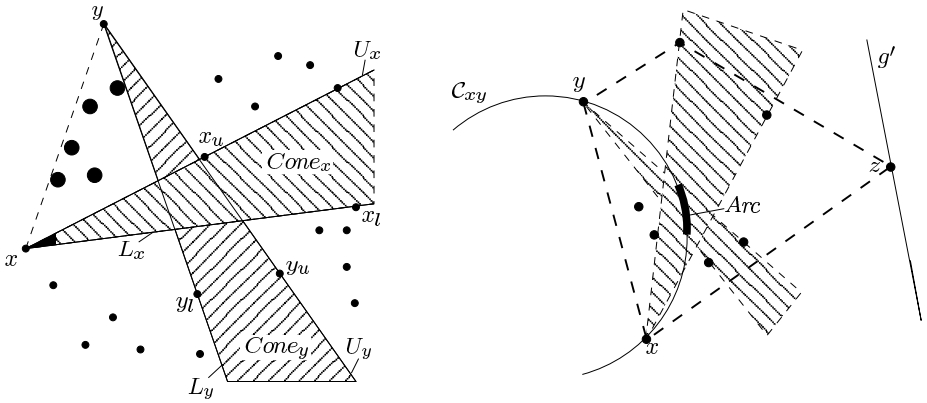

Fig. 5: (a) Determination of $k_{xy} = 5$ between U_x and L_y. (b) Determination of g'.

edges L_y and U_x. This number can be obtained in linear time by comparing the position of every point in P against L_y and U_x. Value k_{yz} can be found analogous.

Finally, the test whether c is the center of equiangularity can be done in linear time like in the algorithm for the case "n is even".

□

4 Conclusions

We have given an algorithm that decides in time $O(n^4 \log n)$ whether n points are in σ-angular configuration, and if so, the algorithm outputs a center of σ-angularity. The adaption of this algorithm for biangular configurations runs in cubic time, if the corresponding two angles are given. Finally, for the case of equiangularity, we have given an algorithm that runs even in time $O(n)$.

While the algorithm for equiangularity is already asymptotically optimal, we believe that the running time of the algorithm for σ-angularity allows to be significantly improved.

As already stated, our algorithm for equiangularity allows to find the Weber point for such configurations, and this is a rather surprising result, since algorithms to compute the Weber point (in finite time) are known for only few other patterns. We are not aware of any general characterization of patterns where the Weber point is easy to find; this might be an interesting line of future research.

References

1. L. Anderegg, M. Cieliebak, G. Prencipe, and P. Widmayer. When is Weber Point Invariant? *(Manuscript)*.

2. C. Bajaj. The Algebraic Degree of Geometric Optimization Problem. *Discrete and Computational Geometry*, 3:177–191, 1988.
3. R. Chandrasekaran and A. Tamir. Algebraic optimization: The Fermat-Weber location problem. *Mathematical Programming*, 46:219–224, 1990.
4. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *Proceedings of ICALP 2003*, pages 1181–1196, 2003.
5. E. J. Cockayne and Z.A. Melzak. Euclidean constructibility in graph-minimization problems. *Math. Magazine*, 42:206 – 208, 1969.
6. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
7. J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 1997.
8. E. Weiszfeld. Sur le Point Pour Lequel la Somme Des Distances de n Points Donnés Est Minimum. *Tohoku Mathematical*, 43:355–386, 1936.