Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

Article

NLP formulation for polygon optimization problems

Saeed Asaeedi [1], Farzad Didehvar [1,*] and Ali Mohades [1]

1 Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran; {asaeedi, didehvar, mohades}@aut.ac.ir

* Correspondence: didehvar@aut.ac.ir; Tel.: +982164545665

Abstract: In this paper, we generalize the problems of finding simple polygons with the minimum area, maximum perimeter and maximum number of vertices so that they contain a given set of points and their angles are bounded by $\alpha + \pi$ where α ($0 \leq \alpha \leq \pi$) is a parameter. We also consider the maximum angle of each possible simple polygon crossing a given set of points, and derive an upper bound for the minimum of these angles. The correspondence between the problems of finding simple polygons with the minimum area and maximum number of vertices is investigated from a theoretical perspective. We formulate the three generalized problems as nonlinear programming models, and then present a Genetic Algorithm to solve them. Finally, the computed solutions are evaluated on several datasets and the results are compared with those from the optimal approach.

Keywords: α-MAP; α-MPP; α-MNP; Polygon optimization; Nonlinear programming; Computational Geometry

---

1. Introduction

Polygons are one of the fundamental objects in the field of computational geometry. The simple polygonization is a way to construct all possible simple polygons on a set of points in the plane. Global optimization problems such as optimal area and perimeter polygonization [1,2] are of major interest to researchers and arising in various application areas such as image processing [3,4], pattern recognition [3,5,6], GIS [7], sensor networks [8,9], etc.

Minimum and maximum perimeter polygonization problems are known as Traveling Salesman Problem (TSP) and Maximum Traveling Salesman Problem (Max-TSP), respectively, which are NP-complete problems [1,10]. Fekete considered the set of points on the grid and showed that the problems of Minimum Area Polygonization (MAP) and Maximum Area Polygonization (MAXP) on that set of points are NP-complete [2]. Recently, it has been shown that computing α-concave hull, as a generalization of MAP, is still NP-hard [11].

To the best of our knowledge, little attention has been paid to the constraint on the internal angles of polygons in previous studies [12–14]. In this paper, we explore the optimum polygonization such that the internal angles of the polygons are bounded. Here, we define α-MAP, α-MPP and α-MNP as the problems of computing simple polygons containing a set of points in the plane with minimum area, maximum perimeter and maximum number of vertex points, respectively, such that all internal angles of the polygons are less than or equal to $\pi + \alpha$. We consider α-MAP, α-MPP and α-MNP as generalizations of computing convex hull and formulate them as nonlinear programming models.

For a set S of points in the plane and for $k \geq 2$, an algorithm for finding k convex polygons that covers S is presented in [15], such that the total area of the convex polygons is minimized. Also, for $k = 2$, another algorithm is presented to minimize the total perimeter of the convex polygons.

There are many NP-complete problems such as TSP [16], packing problems [17], convex shape decomposition [18], and path planning problem [19] that can be formulated as integer programming problems. In [20], an NLP model is presented for the problem of cutting circles from rectangles with minimum area. In [21] the rectangular cartogram problem is formulated as a bilinear program. Raimund Seidel constructs convex hulls of n points in R^d for $d > 3$ using a linear programming algorithm [22].

There are many studies on approximation and randomized algorithms for MAP [11,23,24], MAXP [23], TSP [25,26] and Max-TSP [27,28]. Here, we apply a Genetic Algorithm (GA) to solve α-MAP, α-MPP and α-MNP and then compare the results with those from the optimal approach. GA is used to solve many problems such as TSP [29–31], packing of polygons [32], path planning [33,34], and pattern recognition [35].

χ-shape [36], α-shape [37], Concave hull [7], and crust [38] are all bounding hulls of a set of points as same as the convex hull. α-shape and Concave hull are generalizations of convex hull to cover a set of points and can be used in the fields of space decomposition [39], sensor networks [40], bioinformatics [41], feature detection [42], GIS [43], dataset classification [44], shape reconstruction [45, 46], etc. We implement the Concave hull algorithm [47] and then use the computed results to make comparison against those obtained from the GA.

The rest of the paper is as follows: In section 2, we present some notations and definitions which are required throughout the paper. In section 3, we first formulate the required functions and then introduce nonlinear programming models for α-MAP, α-MPP and α-MNP. In section 4, our theoretical results are discussed. For a set S of points, an upper bound for θ is obtained such that θ is the minimum of maximum angles of each simple polygon containing S. Also, the similarity of the two problems α-MAP and α-MNP are investigated on the grid points. Section 5 is devoted to a full evaluation of our experimental results obtained by implementing the GA and the brute-force algorithm. Section 6 concludes the paper highlighting its main contribution.

2. Preliminaries

Let $S = \{s_1, s_2, ..., s_n\}$ be a set of points in the plane and CH be the convex hull of S. The vertices and edges of CH are denoted by $V_{CH} = \{c_1, c_2, ..., c_m\}$ and $E_{CH} = \{e_1, e_2, ..., e_m\}$, respectively. Furthermore, let $IP = \{a_1, a_2, ..., a_r\}$ be the inner points of CH such that $r = n - m$. A polygon P containing S is specified by a closed chain of vertices $P = (p_1, p_2, ..., p_l, p_1)$. Table 1 shows more notations that are used in the rest of the paper. The simple polygon P contains S iff $V_P \subseteq S$ and $\forall i \in \{1, 2, ..., n\}$, $s_i \in P$. Moreover, P crosses a point x iff $x \in V_P$.

Table 1. Notations of symbols

Notation	Description
S	A set of points in the plane
n	cardinality of S
s_i	ith point of S $(1 \leq i \leq n)$
CH	convex hull of S
m	number of vertices of CH
IP	inner points of CH
V_P	vertices of P
E_P	edges of P
r	cardinality of IP
c_j	jth vertex of CH $(1 \leq j \leq m)$
e_j	jth edge of CH $(1 \leq j \leq m)$
P	a simple Polygon containing S
$\overline{s_i s_j}$	an edge of P with s_i and s_j as its end points $(1 \leq i, j \leq n, i \neq j)$
$\wp(S)$	set of all simple polygons containing S
$Area(P)$	area of polygon P
$Perimeter(P)$	perimeter of polygon P
$Boundary(P)$	number of vertices of P
α	an angle between 0 and π
MAP	problem of computing a simple polygon containing S with minimum area
MPP	problem of computing a simple polygon containing S with maximum perimeter
MNP	problem of computing a simple polygon containing S with maximum number of vertices
CHP	problem of computing convex hull of S
SPP	problem of computing a simple polygon crossing S
$\overset{\frown}{AB}$	the measure of arc AB

MAP is the problem of computing the simple polygon $M \in \wp(S)$ such that $\forall P \in \wp(S), Area(M) \leq Area(P)$, MPP is the problem of computing the simple polygon $E \in \wp(S)$ such that $\forall P \in \wp(S), Perimeter(E) \geq Perimeter(P)$ and MNP is the problem of computing the simple polygon $C \in \wp(S)$ such that $\forall P \in \wp(S), Boundary(C) \geq Boundary(P)$. The following definitions introduce the problems of computing α-MAP, α-MPP and α-MNP.

Definition 1. *For $0 \leq \alpha \leq \pi$, a simple polygon $P \in \wp(S)$ is an α-polygon if all internal angles of P are less than or equal to $\pi + \alpha$ [11].*

Definition 2. *α-MAP, α-MPP and α-MNP are the problems of computing the α-polygon containing S with minimum area, maximum perimeter and maximum number of vertices, respectively.*

In the case of $\alpha = \pi$, the α-polygon, α-MAP, α-MPP and α-MNP will be converted into the simple polygon, MAP, MPP and MNP, respectively. Also, in the case of $\alpha = 0$, the α-polygon will be converted into the convex polygon, and all of the α-MAP, α-MPP and α-MNP will be converted into CHP. We formulate these as binary nonlinear programming models.

Definition 3. *Let $\{c_1, c_2, ..., c_m\}$ be the vertices of CH, $e_j = \overline{c_j c_{j+1}}$ be the jth edge of CH and P be a simple polygon containing S. The points $\{b_{1,j}, b_{2,j}, ..., b_{t,j}\} \in S$ are assigned to the edge e_j if $(c_j, b_{1,j}, b_{2,j}, ..., b_{t,j}, c_{j+1})$ is a chain in P.*

Fig. 1 shows that the polygon P assigns the points $\{b_{1,1}, b_{2,1}, b_{3,1}\}$ to the edge e_1 and the point $\{b_{1,2}\}$ to the edge e_2 and so on. The inner points of P are unassigned. We assume that c_j is assigned to e_j.

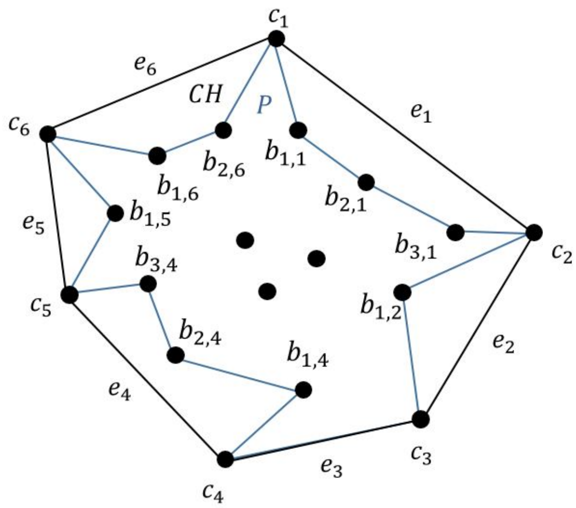

Figure 1. Polygon P assigns internal points to convex hull edges

3. Modeling

In this section we present nonlinear programming models for α-MAP, α-MPP and α-MNP. We first introduce the indices, input data and variables that are used in our models and then formulate the required functions.

3.1. Indices

The following indices are utilized to formulate the problems α-MAP, α-MPP and α-MNP as binary nonlinear programming models.

- $i \in \{1, 2, ..., n\}$ is an index counting the points of S. The point s_{n+1} is identified by s_1 and the point s_{n+2} is identified by s_2.
- $j \in \{1, 2, ..., m\}$ is an index counting the edges in E_{CH} and the vertices in V_{CH}. The edge e_{m+1} is identified by e_1 and the vertex c_{m+1} is identified by c_1.
- $k \in \{0, 1, ..., r\}$ specifies the order of assigned points for an edge of convex hull. $b_{k,j}$ is the kth point which is assigned to e_j. Assume that c_i is assigned to e_i at the position 0.

3.2. Input Data

The input data is as follows:

- n: The number of points in S.
- $(x_i, y_i) \in \mathbb{R}^2$: The coordinate of the point s_i.
- $\alpha \in [0, \pi]$: The constraint for angles.

Assumptions:

- $x^{(j)}$ is the x-coordinate of c_j.
- $y^{(j)}$ is the y-coordinate of c_j.

3.3. Variables

In this model, we have $n.m.r$ variables, denoted by z, which is defined as follows:
$z_{i,j,k}$: A binary variable such that $z_{i,j,k} = 1$ iff the point s_i is assigned to e_j at the position of k.

3.4. Functions

The functions used in this model are listed below.

- $Area(P)$: The area of the polygon P.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

- $Perimeter(P)$: The perimeter of the polygon P.
- $Boundary(P)$: The number of vertices of the polygon P.
- $X(a)$: The x-coordinate of the point a in the plane.
- $Y(a)$: The y-coordinate of the point a in the plane.
- $X(j,k)$: The x-coordinate of the kth points that is assigned to e_j.

$$X(j,k) = \Sigma_{i=1}^{n} z_{i,j,k} \cdot x_i \tag{1}$$

- $Y(j,k)$: The y-coordinate of the kth points that is assigned to e_j.

$$Y(j,k) = \Sigma_{i=1}^{n} z_{i,j,k} \cdot y_i \tag{2}$$

- $Adjust(i_1,i_2)$: If $\exists e \in E_P$ such that s_{i_1} and s_{i_2} are endpoints of e, then $Adjust(i_1,i_2) = 1$, otherwise, $Adjust(i_1,i_2) = 0$.
- $Conflict(i_1,i_2,i_3,i_4)$: If two edges $\overline{s_{i_1}s_{i_2}}$ and $\overline{s_{i_3}s_{i_4}}$ cross each other, then $Conflict(i_1,i_2,i_3,i_4) = 1$, otherwise, $Conflict(i_1,i_2,i_3,i_4) = 0$.
- $Angle(i_1,i_2,i_3)$: The clockwise angle between $\overline{s_{i_1}s_{i_2}}$ and $\overline{s_{i_2}s_{i_3}}$.

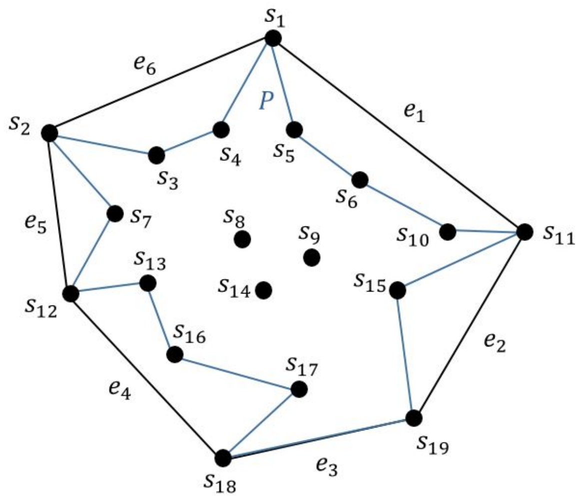

Figure 2. Polygon $P = (s_1, s_5, s_6, ..., s_4, s_1)$ containing $S = \{s_1, ..., s_{19}\}$ assigns internal points to convex hull edges

Fig. 2 is an example of a polygon P containing the set $S = \{s_1, s_2, ..., s_{19}\}$. In this example, the points $\{s_5, s_6, s_{10}\}$ are assigned to e_1. Hence, we have, $Z_{5,1,1} = Z_{6,1,2} = Z_{10,1,3} = 1$. Also, since s_1 is assigned to e_1 at the position 0, $Z_{1,1,0} = 1$. In the same way, for the other edges of CH, we have: $Z_{11,2,0} = Z_{15,2,1} = 1$, $Z_{19,3,0} = 1$, $Z_{18,4,0} = Z_{17,4,1} = Z_{16,4,2} = Z_{13,4,3} = 1$, $Z_{12,5,0} = Z_{7,5,1} = 1$, $Z_{2,6,0} = Z_{3,6,1} = Z_{4,6,2} = 1$. In Fig. 2, to compute $X(j,k)$ for $j = 4$ and $k = 2$ we have:
$X(4,2) = \Sigma_{i=1}^{19} z_{i,4,2} \cdot x_i = 0 + ... + 0 + x_{16} + 0 + 0 + 0 = x_{16}$.

3.5. Models

Here, we present the nonlinear programming formulations for α-MAP, α-MPP and α-MNP. We first formulate the functions $Adjust$, $Conflict$ and $Angle$ as follows:

Adjust function

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

The Adjust function is computed as follows:

$$Adjust(i_1,i_2)=\begin{cases}1 & \exists k\in\{0,...,r-1\},\exists j\in\{1,...,m\}\mid \quad z_{i_1,j,k}=z_{i_2,j,k+1}=1\\ 1 & \exists k\in\{0,...,r-1\},\exists j\in\{1,...,m\}\mid \quad z_{i_2,j,k}=z_{i_1,j,k+1}=1\\ 1 & \exists k\in\{1,...,r\},\exists j\in\{1,...,m\}\mid \quad z_{i_1,j,k}=z_{i_2,j+1,0}=1,\\ & \qquad \forall i\in\{1,...,n\}z_{i,j,k+1}=0\\ 1 & \exists k\in\{1,...,r\},\exists j\in\{1,...,m\}\mid \quad z_{i_2,j,k}=z_{i_1,j+1,0}=1,\\ & \qquad \forall i\in\{1,...,n\}z_{i,j,k+1}=0\\ 0 & \text{o.w.}\end{cases} \tag{3}$$

As seen in Fig. 2, s_6 is adjusting to s_{10}. Since $Z_{6,1,2}=Z_{10,1,3}=1$, we have $Adjust(6,10)=1$. Also, since $z_{10,1,3}=z_{11,2,0}=1$ and $\forall i\in\{1,...,n\}z_{i,1,4}=0$, we have $Adjust(10,11)=1$.

Conflict function

To compute the conflict function, consider the following expression:

$$\begin{aligned} E_{i_1,i_2} &= (X(s_{i_2})-X(s_{i_1}),Y(s_{i_2})-Y(s_{i_1}))\\ R_{i_1,i_2} &= (-Y(E_{i_1,i_2}),X(E_{i_1,i_2}))\\ h(i_1,i_2,i_3,i_4) &= (E_{i_3,i_1}.R_{i_1,i_2})/(E_{i_3,i_4}.R_{i_1,i_2})\\ &= \frac{(X(E_{i_3,i_1})\cdot X(R_{i_1,i_2})+Y(E_{i_3,i_1})\cdot Y(R_{i_1,i_2}))}{(X(E_{i_3,i_4})\cdot X(R_{i_1,i_2})+Y(E_{i_3,i_4})\cdot Y(R_{i_1,i_2}))}\end{aligned} \tag{4}$$

So, the function $Conflict(i_1,i_2,i_3,i_4)$ is computed as follows:

$$Conflict(i_1,i_2,i_3,i_4)=\begin{cases} & 0\le h(i_1,i_2,i_3,i_4)\le 1 \quad and\\ 1 & Adjust(i_1,i_2)=Adjust(i_3,i_4)=1 \quad and\\ & Adjust(i_1,i_3)=Adjust(i_2,i_4)=0\\ 0 & \text{o.w.}\end{cases} \tag{5}$$

Based on the mentioned notations, P is simple if $\forall i_1,i_2,i_3,i_4\in\{1,2,...,n\}$ such that $i_1\neq i_2\neq i_3\neq i_4$, $Adjust(i_1,i_2)=Adjust(i_3,i_4)=1$ and $Adjust(i_1,i_3)=Adjust(i_2,i_4)=0$, we have $Conflict(i_1,i_2,i_3,i_4)=0$.

Angle function

The polygon P is an α-polygon iff $\forall i_1,i_2,i_3\in\{1,2,...,n\}$ such that $i_1\neq i_2\neq i_3$, $Adjust(i_1,i_2)=Adjust(i_2,i_3)=1$ and $Adjust(i_1,i_3)=0$, we have $Angle(i_1,i_2,i_3)\le\pi+\alpha$. The angle between two line segments A and B can be computed as follows:

$$\theta=\arccos\frac{A.B}{(|A|\cdot|B|)} \tag{6}$$

Based on the mentioned notations, let

$$\begin{aligned} A_{i_1,i_2} &= (X(s_{i_2})-X(s_{i_1}),Y(s_{i_2})-Y(s_{i_1}))\\ B_{i_2,i_3} &= (X(s_{i_3})-X(s_{i_2}),Y(s_{i_3})-Y(s_{i_2}))\end{aligned} \tag{7}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

If $Adjust(i_1, i_2) = Adjust(i_2, i_3) = 1$ and $Adjust(i_1, i_3) = 0$, $Angle(i_1, i_2, i_3)$ is computed as follows, otherwise $Angle(i_1, i_2, i_3) = 0$.

$$Angle(i_1, i_2, i_3) = \arccos \frac{X(A_{i_1,i_2}) \cdot X(B_{i_2,i_3}) + Y(A_{i_1,i_2}) \cdot Y(B_{i_2,i_3})}{\sqrt{X(A_{i_1,i_2})^2 + Y(A_{i_1,i_2})^2} \cdot \sqrt{X(B_{i_2,i_3})^2 + Y(B_{i_2,i_3})^2}} \quad (8)$$

3.5.1. Modeling α-MAP

α-MAP is the problem of computing the α-polygon with the minimum area on a set of points. Since $\wp(S)$ is the set of all simple polygons containing S, α-MAP can be formulated as follows:

$$\begin{aligned} & \min_{P \in \wp(S)} Area(P) \\ s.t. \quad & \\ & \text{All internal angles of } P \text{ are less than or equal to } \pi + \alpha \end{aligned} \quad (9)$$

As seen in Fig. 1, each polygon $P \in \wp(S)$ assigns the points of S to the edges of CH. Therefore, each simple polygon containing S is equivalent to an assignments of the points of S to the edges of CH, and each assignment is determined by an evaluation of $z_{i,j,k}$ for all i, j, k. In the following, the area function is formulated as objective function of the model.

Theorem 1. *Let $P \in \wp(S)$ be a simple polygon and Z be the corresponding assignment for P. The area of P is computed as follows:*

$$\begin{aligned} & Area(P) = \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \\ & [[\Sigma_{i=1}^{n} z_{i,j,k} \cdot x_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k})) \cdot x^{(j+1)} + \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot x_i] \times \\ & [\Sigma_{i=1}^{n} z_{i,j,k} \cdot y_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k})) \cdot y^{(j+1)} - \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot y_i]] \end{aligned} \quad (10)$$

Proof. Based on the Shoelace formula (also known as the surveyor's formula [48]), the area of a polygon $P = (p_1, p_2, ..., p_l, p_1)$ is:

$$Area(P) = \Sigma_{i=1}^{l} (X(p_{i+1}) + X(p_i)) \cdot (Y(p_{i+1}) - Y(p_i)) \quad (11)$$

As an example, assume that P is the polygon of Fig. 2. So, $p_1 = s_1$, $p_2 = s_5$, $p_3 = s_6$, $p_4 = s_{10}$, and $p_5 = s_{11}$. Let $T_i = (X(p_{i+1}) + X(p_i)) \cdot (Y(p_{i+1}) - Y(p_i))$, thus we have:

$$T_1 = (X(p_2) + X(p_1)) \cdot (Y(p_2) - Y(p_1)) = (X(s_5) + X(s_1)) \cdot (Y(s_5) - Y(s_1)) \quad (12)$$

Since the points s_5 and s_1 are assigned to e_1 at positions 1 and 0 respectively, we have:

$$T_1 = (X(1, 1) + X(1, 0)) \cdot (Y(1, 1) - Y(1, 0)) \quad (13)$$

In the same way,

$$\begin{aligned} T_2 &= (X(p_3) + X(p_2)) \cdot (Y(p_3) - Y(p_2)) = (X(1, 2) + X(1, 1)) \cdot (Y(1, 2) - Y(1, 1)) \\ T_3 &= (X(p_4) + X(p_3)) \cdot (Y(p_4) - Y(p_3)) = (X(1, 3) + X(1, 2)) \cdot (Y(1, 3) - Y(1, 2)) \\ T_4 &= (X(p_5) + X(p_4)) \cdot (Y(p_5) - Y(p_4)) = (X(2, 0) + X(1, 3)) \cdot (Y(2, 0) - Y(1, 3)) \end{aligned} \quad (14)$$

Based on the above equations, we employ the bellow formula for T_1,T_2 and T_3 so that:

$$T_k = (X(1, k) + X(1, k - 1)) \cdot (Y(1, k) - Y(1, k - 1)) \quad (15)$$

Equation (15) cannot be used for T_4:
$(X(1, 4) + X(1, 3)) \cdot (Y(1, 4) - Y(1, 3)) \neq (X(2, 0) + X(1, 3)) \cdot (Y(2, 0) - Y(1, 3))$

In other words, equation (15) can be used while the points are assigned to the same edge. In equation (14), for T_4, the point $p_5 = s_{11}$ is assigned to e_2 while the point p_4 is assigned to e_1. Based on equation (1), since $\forall i \in \{1, 2, ..., n\}, z_{i,1,4} = 0 \Rightarrow X(1,4) = \Sigma_{i=1}^{n} z_{i,1,4} = 0 \Rightarrow (1 - \Sigma_{i=1}^{n} z_{i,1,4}) \cdot x^{(2)} = x^{(2)} \Rightarrow (1 - \Sigma_{i=1}^{n} z_{i,1,4}) \cdot x^{(2)} = X(p_5)$. Hence, $(1 - \Sigma_{i=1}^{n} z_{i,1,4}) \cdot x^{(2)}$ can be used to compute $X(p_5)$.

$$\begin{aligned} T_k = \; & (X(1,k) + (1 - \Sigma_{i=1}^{n} z_{i,1,k}) \cdot x^{(2)} + X(1,k-1)) \times \\ & (Y(1,k) + (1 - \Sigma_{i=1}^{n} z_{i,1,k}) \cdot y^{(2)} - Y(1,k-1)) \end{aligned} \tag{16}$$

Based on equation (16):

$$\begin{aligned} T_4 = \; & (X(1,4) + (1 - \Sigma_{i=1}^{n} z_{i,1,4}) \cdot x^{(2)} + X(1,3)) \times \\ & (Y(1,4) + (1 - \Sigma_{i=1}^{n} z_{i,1,4}) \cdot y^{(2)} - Y(1,3)) \\ T_4 = \; & (0 + (1-0)x^{(2)} + X(1,3)) \cdot (0 + (1-0)y^{(2)} - Y(1,3)) \\ T_4 = \; & (X(2,0) + X(1,3)) \cdot (Y(2,0) - Y(1,3)) \end{aligned} \tag{17}$$

Since based on equation (1), X(1,k) is equal to 0 for all $k \geq 4$, we have:

$$\begin{aligned} T_5 &= (0 + (1-0)x^{(2)} + X(1,4)) \cdot (0 + (1-0)y^{(2)} - Y(1,4)) \\ T_5 &= (0 + X(2,0) + 0) \cdot (0 + Y(2,0) - 0) \end{aligned} \tag{18}$$

Hence, in order to avoid extra summation, we employ the following equation:

$$\begin{aligned} T_k = \; & [X(1,k) + (\Sigma_{i=1}^{n} z_{i,1,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,1,k}) \cdot x^{(2)} + X(1,k-1)] \times \\ & [Y(1,k) + (\Sigma_{i=1}^{n} z_{i,1,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,1,k}) \cdot y^{(2)} - Y(1,k-1)] \end{aligned} \tag{19}$$

From equation (19), for all $k \geq 5$ we have:

$$T_k = (0 + (0) \cdot (1-0) \cdot x^{(2)} + 0) \times (0 + (0) \cdot (1-0) \cdot y^{(2)} - 0) = 0 \tag{20}$$

Considering the points that are assigned to e_j, equation (19) can be extended as follows:

$$\begin{aligned} T_{j,k} = \; & [X(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} + X(j,k-1)] \times \\ & [Y(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - Y(j,k-1)] \end{aligned} \tag{21}$$

Based on equation (11), the area of the polygon P is computed as follows:

$$\begin{aligned} Area(P) \; = \; & \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} T_{j,k} = \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \\ & [[X(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} + X(j,k-1)] \times \\ & [Y(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - Y(j,k-1)]] \end{aligned} \tag{22}$$

From equations (1), (2) and (22), we have:

$$\begin{aligned} & Area(P) = \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \\ & [[\Sigma_{i=1}^{n} z_{i,j,k} \cdot x_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} + \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot x_i] \times \\ & [\Sigma_{i=1}^{n} z_{i,j,k} \cdot y_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot y_i]] \end{aligned} \tag{23}$$

□

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

Based on Theorem 1, equation (9) is formulated as follows:

$$
\begin{aligned}
&\min \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \\
&[[\Sigma_{i=1}^{n} z_{i,j,k} \cdot x_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} + \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot x_i] \times \\
&[\Sigma_{i=1}^{n} z_{i,j,k} \cdot y_i + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - \Sigma_{i=1}^{n} z_{i,j,k-1} \cdot y_i]] \\
s.t.& \\
&z_{i,j,k} \in \{0,1\} \qquad \forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}, \forall k \in \{1,...,r\} \qquad (1) \\
&\Sigma_{j=1}^{m} \Sigma_{k=1}^{r} z_{i,j,k} \leq 1 \qquad \forall i \in \{1,2,...,n\} \qquad (2) \\
&conflict(i_1,i_2,i_3,i_4) = 0 \qquad \forall i_1,i_2,i_3,i_4 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 \neq i_4 \qquad (3) \\
&angle(i_1,i_2,i_3) \leq \pi + \alpha \qquad \forall i_1,i_2,i_3 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 \qquad (4)
\end{aligned} \tag{24}
$$

In equation (24), constraint (1) considers all assignments of the points while constraint (2) prevents assigning a point to more than one edge. The point s_i is unassigned if $\Sigma_{j=1}^{m} \Sigma_{k=1}^{r} z_{i,j,k} = 0$, and assigned to one edge if $\Sigma_{j=1}^{m} \Sigma_{k=1}^{r} z_{i,j,k} = 1$. Also, constraint (2) prevents assigning a point to more than one position on an edge. In addition, constraint (3) guarantees that the constructed polygon is simple while constraint (4) ensures that it is an α-polygon.

When $\alpha = 0$, the solution of equation (24) is an assignment that constructs the convex hull of the points and when $\alpha = \pi$, the solution of equation (24) is an assignment that constructs M as the solution of MAP on the points. There is an algorithm to compute CH in $O(n \log n)$ time [49], while MAP is NP-complete.

Fig. 3 illustrates the solution of α-MAP on a set of points for different values of α.

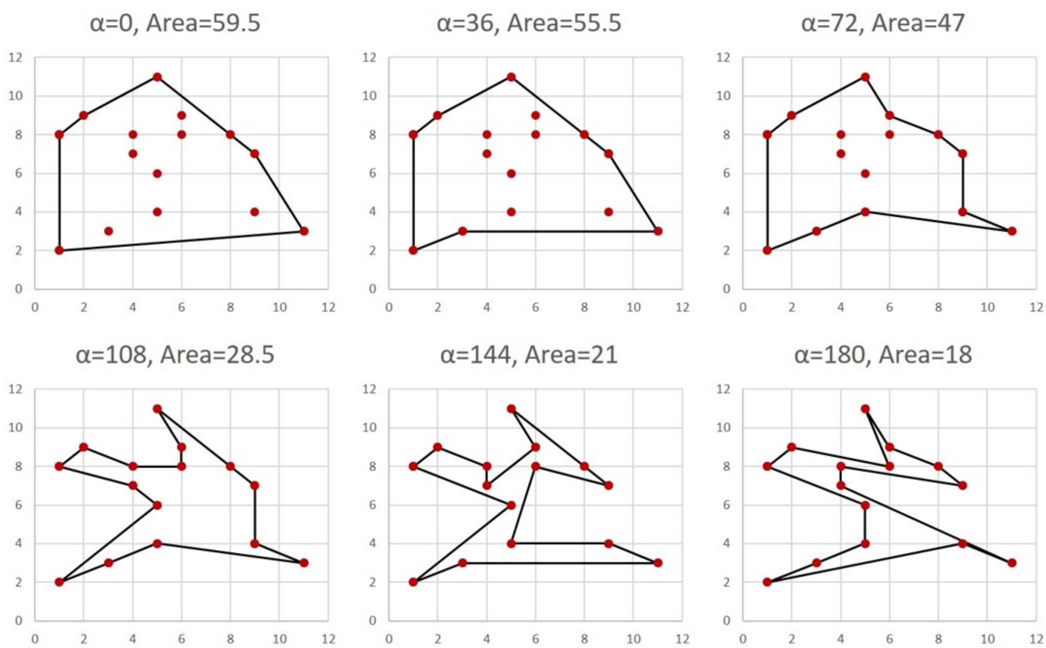

Figure 3. Solution of α-MAP for different values of α

3.5.2. Modeling α-MPP

α-MPP is the problem of computing the α-polygon with the maximum perimeter on a set of points. Since $\wp(S)$ is the set of all simple polygons containing S, α-MPP is computed as follows:

$$
\begin{aligned}
&\max_{P \in \wp(S)} \; Perimeter(P) \\
s.t.& \\
&\text{All internal angles of } P \text{ are less than or equal to } \pi + \alpha
\end{aligned} \tag{25}
$$

Let $P = (p_1, p_2, ..., p_l, p_1)$ be a polygon containing S. The perimeter of P is the total length of its edges:

$$Perimeter(P) = \Sigma_{i=1}^{l} \sqrt{(X(p_{i+1}) - X(p_i))^2 + (Y(p_{i+1}) - Y(p_i))^2} \tag{26}$$

By using Z as the corresponding assignment for P, similar to Theorem 1, the perimeter of P is computed as follows:

$$Perimeter(P) = \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \sqrt{\begin{array}{l}(X(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} - X(j,k-1))^2 + \\ (Y(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - Y(j,k-1))^2\end{array}} \tag{27}$$

Based on equations (25) and (27), we have the following formula for α-MPP:

$$\begin{array}{ll} \max \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} \sqrt{\begin{array}{l}(X(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot x^{(j+1)} - X(j,k-1))^2 + \\ (Y(j,k) + (\Sigma_{i=1}^{n} z_{i,j,k-1}) \cdot (1 - \Sigma_{i=1}^{n} z_{i,j,k}) \cdot y^{(j+1)} - Y(j,k-1))^2\end{array}} & \\ s.t. & \\ \quad z_{i,j,k} \in \{0,1\} \quad \forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}, \forall k \in \{1,...,r\} & (1) \\ \quad \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} z_{i,j,k} \leq 1 \quad \forall i \in \{1,2,...,n\} & (2) \\ \quad conflict(i_1,i_2,i_3,i_4) = 0 \quad \forall i_1,i_2,i_3,i_4 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 \neq i_4 & (3) \\ \quad angle(i_1,i_2,i_3) \leq \pi + \alpha \quad \forall i_1,i_2,i_3 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 & (4) \end{array} \tag{28}$$

When $\alpha = 0$, the solution of equation (28) is an assignment that constructs the convex hull of the points and when $\alpha = \pi$, it is an assignment that constructs E as the solution of MPP, which is known as Max-TSP, on the points. There is an algorithm to compute CH in $O(n \log n)$ time, while Max-TSP is a well-known NP-complete problem.

Fig. 4 illustrates the solution of α-MPP on a set of points for different values of α.

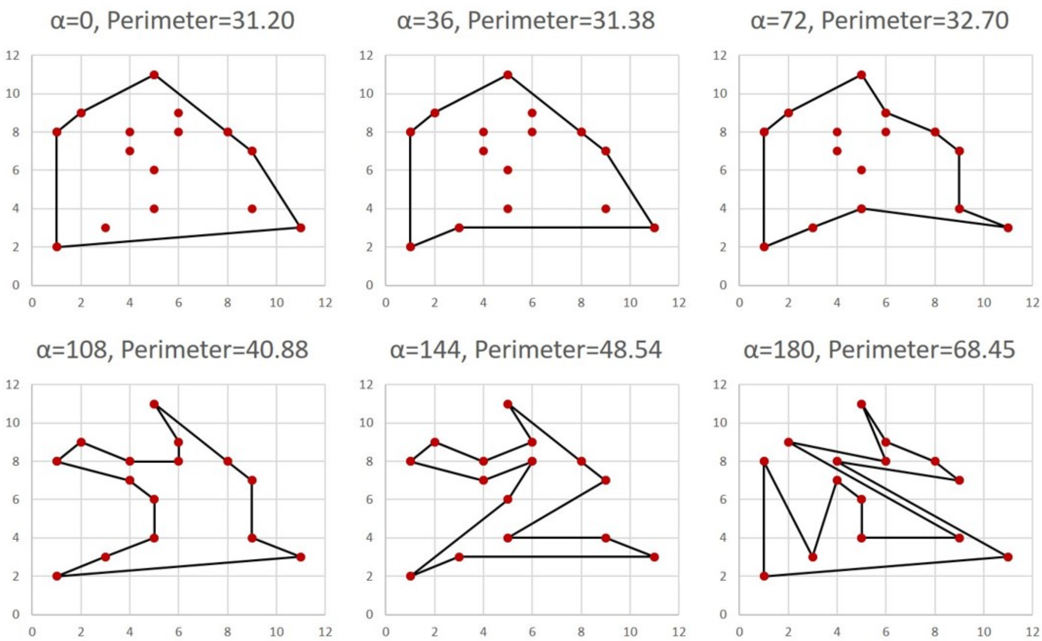

Figure 4. Solution of α-MPP for different values of α

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

3.5.3. Modeling α-MNP

α-MNP on a set of points is the problem of computing the α-polygon with the maximum number of vertices. Since $\wp(S)$ is the set of all simple polygons containing S, α-MNP is computed as follows:

$$\begin{aligned} & \max_{P \in \wp(S)} Boundary(P) \\ s.t. \quad & \\ & \text{All internal angles of } P \text{ are less than or equal to } \pi + \alpha \end{aligned} \tag{29}$$

As stated before, $z_{i,j,k}$ is equal to 1 iff the point s_i is assigned to the edge e_j at the position k. Hence, the following equation specifies the number of vertex points for the constructed polygon P:

$$Boundary(P) = \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} \Sigma_{k=0}^{r} z_{i,j,k} \tag{30}$$

Similar to α-MAP and α-MPP, α-MNP is formulated as follows:

$$\begin{aligned} & \max \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} \Sigma_{k=0}^{r} z_{i,j,k} \\ s.t. \quad & \\ & z_{i,j,k} \in \{0,1\} \qquad \forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}, \forall k \in \{1,...,r\} && (1) \\ & \Sigma_{j=1}^{m} \Sigma_{k=1}^{r} z_{i,j,k} \leq 1 \qquad \forall i \in \{1,2,...,n\} && (2) \\ & conflict(i_1,i_2,i_3,i_4) = 0 \qquad \forall i_1,i_2,i_3,i_4 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 \neq i_4 && (3) \\ & angle(i_1,i_2,i_3) \leq \pi + \alpha \qquad \forall i_1,i_2,i_3 \in \{1,2,...,n\} \mid i_1 \neq i_2 \neq i_3 && (4) \end{aligned} \tag{31}$$

When $\alpha = 0$, the solution of equation (31) is an assignment that constructs the convex hull of the points and when $\alpha = \pi$, the solution of equation (31) is an assignment that constructs C as the solution of MNP on the points. C is a simple polygon that crosses all points. There are optimal algorithms to compute CH and C in $O(n \log n)$ time

Fig. 5 illustrates the solution of α-MNP on a set of points for different values of α.

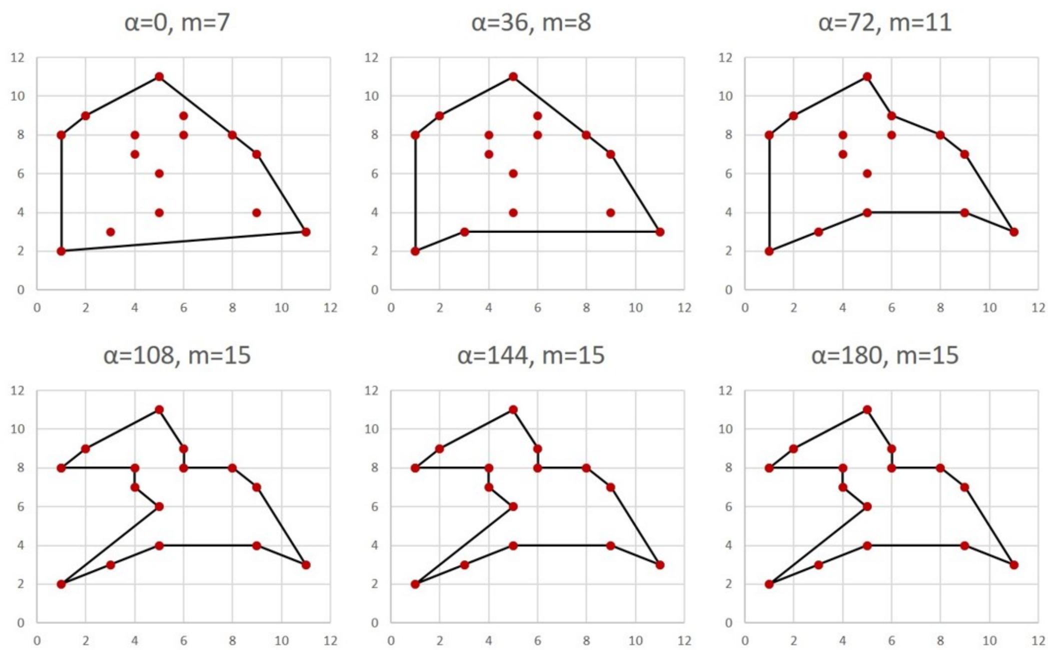

Figure 5. Solution of α-MNP for different values of α

doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

4. Theoretical results

In this section we present our theoretical results. As stated before, α-MNP will be converted into CHP and SPP when $\alpha = 0$ and $\alpha = \pi$, respectively which are solvable in polynomial time. When $\alpha = \pi$, the constructed polygon crosses all points. For each set S of points, the smallest value of α such that α-MNP crosses S is computed in the next subsection.

4.1. Upper bound for α in α-MNP

For each polygon $P \in \wp(S)$, assume that γ_P is the maximum angle of the polygon P. Let θ be the minimum value of γ_P over all $P \in \wp(S)$ that crosses S. For all $\alpha \geq \theta - \pi$, there always exists an α-polygon that crosses S. In other words, the polygon P' such that $\gamma_{P'} = \theta$, satisfies α-MNP for all $\alpha \geq \theta - \pi$. Here, we present an upper bound for θ for any set of points.

In Theorem 2, it is shown that $2\pi - \frac{2\pi}{2^{r-1}m}$ can be interpreted as an upper bound of θ and in Theorem 3 this bound is improved. In the following, we design an algorithm to construct a simple polygon containing S which satisfies these bounds. Let us first define the concept of *"sweep arc"* and then prove some lemmas.

Definition 4. *Let $e = \overline{AB}$ be an edge of polygon P. A sweep arc on the edge e is a minor arc AB where $\widehat{AB} = 0$ and expands to the major arc AB where $\widehat{AB} = \pi$. The direction of expansion is to the inside of the polygon. Fig. 6 depicts the sweep arc on the edge e.*

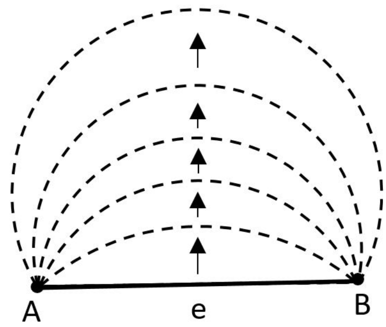

Figure 6. A sweep arc on the edge e

Let e_j be an edge of the polygon P. We denote the major segment with length of β that corresponds to e_j by M_j^β.

Lemma 1. *Let x be a point inside the convex polygon P, $E = \{e_1, e_2, ..., e_m\}$ be the edges of P and $\beta = 2\pi - \frac{4\pi}{m}$. Then, $\exists j \in \{1, 2, ..., m\}$ such that $x \in M_j^\beta$.*

Proof. Let β_j be the angle subtended by e_j at the point x and β_M be the maximum one. Let e be the edge that corresponds to β_M. Since $\Sigma_{j=1}^{m} \beta_j = 2\pi$, we have $\beta_M \geq \frac{2\pi}{m}$ and the corresponding arc of β_M is more than $\frac{4\pi}{m}$. Hence, the measure of sweep arc on the edge e at x is less than $2\pi - \frac{4\pi}{m}$, (see Fig. 7). □

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

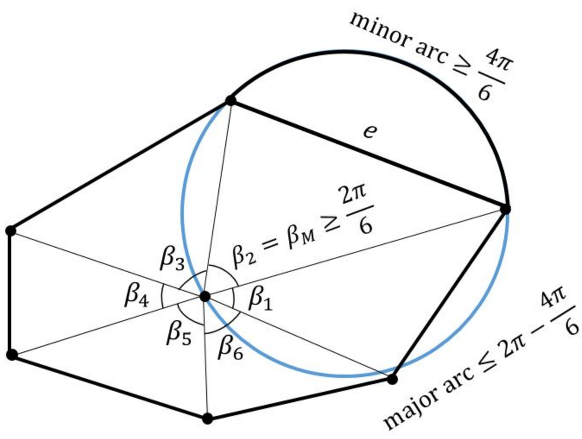

Figure 7. Measure of sweep arc on the edge e is less than $2\pi - \frac{4\pi}{6}$

Lemma 2. *Let P be a convex polygon, $\{e_1, e_2, ..., e_m\}$ be the edges of P and $\beta_{max} = 2\pi - \frac{4\pi}{m}$. The entire P is covered by all major segments with length of β_{max} that correspond to the edges of P, i.e. $P \subset \cup_{j=1}^{m} M_j^{\beta_{max}}$, (see Fig. 8).*

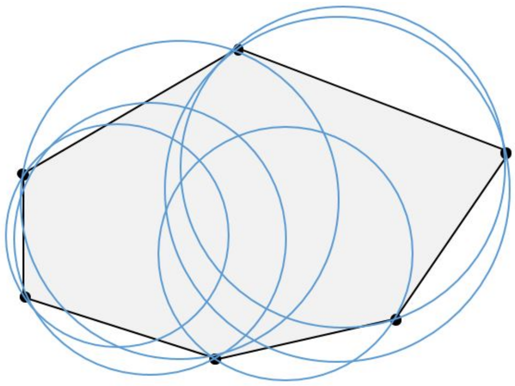

Figure 8. Measure of all major segments are $\beta_{max} = 2\pi - \frac{4\pi}{6}$

Proof. To prove the lemma by reductio ad absurdum, suppose that there exists a point x inside the polygon P and outside of all the major segments. Since the measure of all major segments are equal to $2\pi - \frac{4\pi}{m}$, there is no edge e such that the sweep arc on e touches x at the measure $\beta \leq 2\pi - \frac{4\pi}{m}$, i.e. $\forall j \in \{1, 2, ..., m\}$, $x \notin M_j^{\beta_{max}}$. This contradicts Lemma 1. □

Remark 1. *Suppose that the convex hull of S has $n - 1$ edges, i.e. one point is inside the convex hull. Based on Lemma 2, $2\pi - \frac{2\pi}{n-1}$ is an upper bound for θ over all simple polygons containing S. It is noteworthy that this bound is tight. The tightness is achieved when the inner point is at the center of a regular n-gons as illustrated in Fig. 9.*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

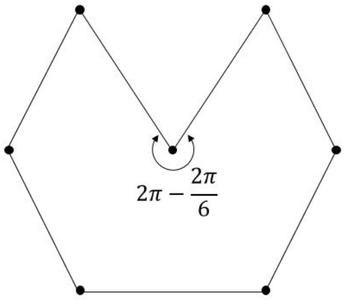

Figure 9. Maximum angle of each polygon containing these points is equal to $2\pi - \frac{2\pi}{6}$

In the following, we generalize the upper bound for any set S of points and then present an algorithm to generate a polygon containing S that satisfies the generalized upper bound. But let us first consider a sweep arc on an edge to measure β_{max} that includes a set of n points.

Lemma 3. *Let $e = \overline{c_1c_2}$ be a line segment and S be a set of n points inside the major segment corresponding to e such that the measure of major arc is $\beta_{max} = 2\pi - \frac{4\pi}{m}$ for an integer number m (see Fig. 10.a). There exists a chain $(s_1, s_2, ..., s_n)$ on S such that all internal angles of $\hat{s}_i$ in the polygon $(c_1, s_1, s_2, ..., s_n, c_2, c_1)$ are greater than or equal to $\frac{2\pi}{2^{n-1}\cdot m}$ (see Fig. 10.b).*

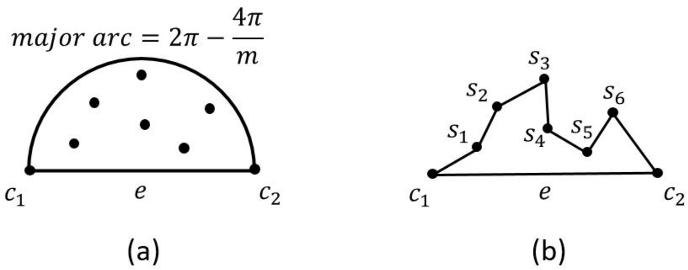

Figure 10. a. Set of 6 points inside the major segment b. All internal angles of the polygon $(c_1, s_1, s_2, s_3, s_4, s_5, s_6, c_2, c_1)$ are greater than or equal to $\frac{2\pi}{32m}$

Proof. Here, we employ the sweep arc algorithm to construct the polygon.

Algorithm 1 (Sweep Arc Algorithm)

Let us sweep the arc from measure 0 to β_{max} on $e = \overline{c_1c_2}$. By so doing, the polygon is constructed, while the arc hits the points. In the following we show how to construct the polygon step by step.

On the first hit:

Let x_1 be the first point that the sweeping arc meets. We construct the polygon by connecting x_1 to c_1 and c_2. Since the maximum measure of the arc is β_{max}, the internal angle of $\hat{x}_1$ in the triangle $(c_1x_1c_2)$ is greater than or equal to $\frac{2\pi}{m}$ (see Fig. 11).

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

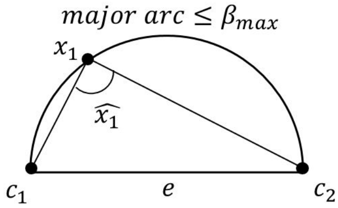

Figure 11. Angle of $\hat{x}_1$ is greater than or equal to $\frac{2\pi}{m}$

On the second hit:

Let x_1 be the first point that the sweeping arc meets and x_2 be the second one. Also, let $e_1 = \overline{c_1x_1}$ and $e_2 = \overline{c_2x_1}$ be two constructed edges in the previous step. e_1 and e_2 divide the sweeping arc into 3 parts; the arc B_1 where e_1 is visible but e_2 is not visible from all the points on it; the arc B_2 where e_2 is visible but e_1 is not visible from all the points on it, and finally the arc B_3 where e_1 and e_2 are visible from all the points on it (see Fig. 12).

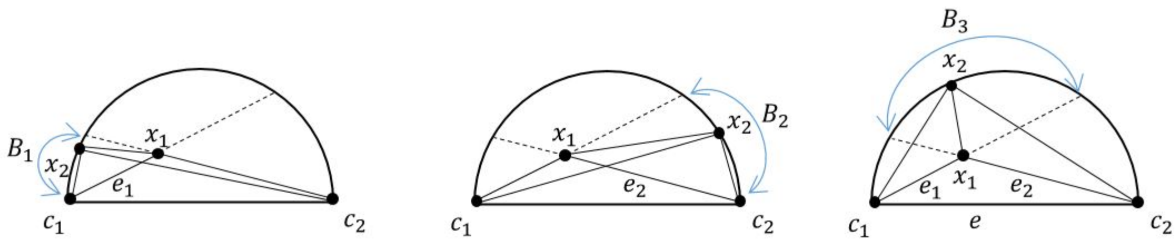

Figure 12. Sweeping arc is divided into 3 parts B_1, B_2 and B_3

Case 1.

If x_2 is placed on B_1: The angle $\widehat{c_1x_2x_1}$ is greater than $\widehat{c_1x_2c_2}$ and the angle $\widehat{c_1x_2c_2}$ is greater than or equal to $\frac{2\pi}{m}$. Hence, the angle $\widehat{c_1x_2x_1}$ is greater than $\frac{2\pi}{m}$. Since the internal angles $\hat{x}_2$ and $\hat{x}_1$ are greater than $\frac{2\pi}{2m}$, we consider the polygon $(c_1x_2x_1c_2c_1)$ as the constructed polygon.

Case 2.

If x_2 is placed on B_2: Based on the same reason mentioned above, the angle $\widehat{c_2x_2x_1}$ is greater than $\frac{2\pi}{m}$. Since the internal angles $\hat{x}_2$ and $\hat{x}_1$ are greater than $\frac{2\pi}{m}$, we consider the polygon $(c_1x_1x_2c_2c_1)$ as the constructed polygon.

Case 3.

If x_2 is placed on B_3: In contrast to the previous cases, the angles $\widehat{c_1x_2x_1}$ and $\widehat{c_2x_2x_1}$ are less than $\widehat{c_1x_2c_2}$, but the maximum of $\widehat{c_1x_2x_1}$ and $\widehat{c_2x_2x_1}$ is greater than $\frac{\widehat{c_1x_2c_2}}{2}$. Since $\widehat{c_1x_2c_2}$ is greater than $\frac{2\pi}{m}$, the maximum of $\widehat{c_1x_2x_1}$ and $\widehat{c_2x_2x_1}$ is greater than $\frac{2\pi}{2m}$. Hence, if $\widehat{c_1x_2x_1} > \widehat{c_2x_2x_1}$, the constructed polygon is $(c_1x_2x_1c_2c_1)$, otherwise, it is $(c_1x_1x_2c_2c_1)$.

In other words, the angular bisector of $\widehat{c_1x_1c_2}$ divides the sweeping arc into 2 parts A_1 and A_2 (see Fig. 13). Any point x_2 on A_1 constructs the angle $\widehat{c_1x_2x_1}$ greater than $\frac{2\pi}{2m}$ and on A_2 constructs the angle $\widehat{x_1x_2c_2}$ greater than $\frac{2\pi}{2m}$. Hence, in the case where point x_2 is placed on A_1, we consider the polygon $(c_1x_2x_1c_2c_1)$ as the constructed polygon and if placed on A_2, we consider the polygon $(c_1x_1x_2c_2c_1)$ as the constructed polygon.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

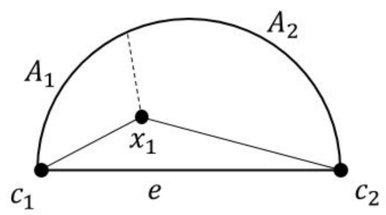

Figure 13. Sweep arc is divided into 2 parts A_1 and A_2

On the third hit:

Without loss of generality, assume that $(c_1x_2x_1c_2c_1)$ is the polygon obtained at the end of the previous step. The angular bisector of $\widehat{c_1x_2x_1}$ divides A_1 into 2 parts A_{11} and A_{12}. Hence, the sweeping arc is divided into 3 parts A_2, A_{11} and A_{12} (see Fig. 14).

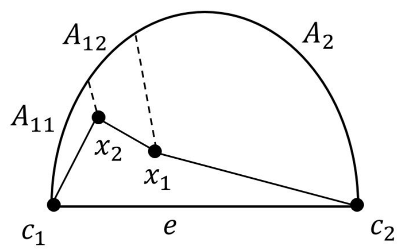

Figure 14. Sweep arc is divided into 3 parts A_2, A_{11} and A_{12}

Based on the previous step, any point x_3 on A_2 leads to the construction of the angle $\widehat{x_1x_3c_2}$ which is greater than $\frac{2\pi}{2m}$. Similarly, any point x_3 on A_{11} and A_{12} leads to the construction of the angles $\widehat{c_1x_3x_2}$ and $\widehat{x_2x_3x_1}$, respectively, which are greater than $\frac{\widehat{c_1x_3x_1}}{2}$. Since any angle $\widehat{c_1x_3x_1}$ on A_1 is greater than $\frac{2\pi}{2m}$, either the angle $\widehat{c_1x_3x_2}$ or $\widehat{x_2x_3x_1}$ is greater than $\frac{2\pi}{4m}$.

Let x_3 be the third point that the sweeping arc meets. If x_3 is placed on A_2 , or on A_{11} or on A_{12}, we consider $(c_1x_2x_1x_3c_2c_1)$ or $(c_1x_3x_2x_1c_2c_1)$ or $(c_1x_2x_3x_1c_2c_1)$ as the constructed polygon, respectively (see Fig. 15).

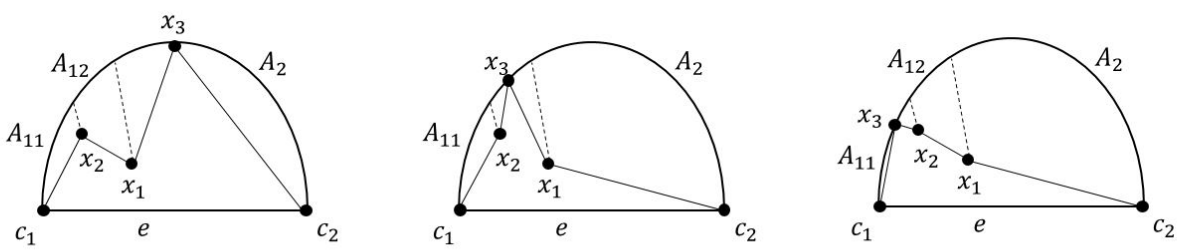

Figure 15. Angles of all constructed simple polygons are greater than or equal to $\frac{2\pi}{4m}$

Generalization:

Assume that $(c_1x_1x_2...x_{n-1}c_2c_1)$ is the obtained polygon at the end of the previous step. Let x_n be the next point touched by the sweeping arc which is divided into n parts $A_1, A_2, ..., A_n$. Considering the worst case, x_n is placed on A_i or on A_{i+1} such that the angular bisector of $\hat{x_i}$ divides the corresponding part into A_i and A_{i+1}. Any point x_n on A_i or on A_{i+1} leads to the construction of the angle $\hat{x_n}$ which is greater than $\frac{\hat{x_i}}{2}$. Based on the previous step and considering the worst case, the angle $\hat{x_i}$ is greater than $\frac{2\pi}{2^{n-2}m}$. Hence, the angle $\hat{x_n}$ is greater than $\frac{2\pi}{2^{n-1}m}$. If x_n is placed on A_1, or on A_2, ... or on A_n, we consider the polygon $(c_1x_nx_1...x_{n-1}c_2c_1)$, or $(c_1x_1x_nx_2...x_{n-1}c_2c_1)$, ... or $(c_1x_1...x_{n-1}x_nc_2c_1)$ as the constructed polygon, respectively. □

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

We refer to the constructed polygon by algorithm 1, as a polygon corresponding to the line segment e. In the following, based on the Lemma 3, we present an algorithm to generate a polygon containing a given set of points such that all internal angles are less than $2\pi - \frac{2\pi}{2^{r-1}m}$.

Theorem 2. *There exists a polygon $P \in \wp(S)$ that crosses S in which all internal angles of P are less than $2\pi - \frac{2\pi}{2^{r-1}m}$.*

Proof. Here, by presenting algorithm 2 we construct the polygon.

Algorithm 2

1. Compute CH as the convex hull of S and let IP be the set of inner points of CH.
2. For each edge e_j of CH:
   (a) Compute the polygon P_j corresponding to the edge e_j using algorithm 1 to meet points of IP.
   (b) Remove vertices of P_j from IP.
3. For all $j \in \{1, 2, ..., m\}$, the edges of P_j minus all edges of CH except those that have no corresponding polygon, construct the desired polygon.

Based on Lemma 2, the entire CH is covered by all major segments that correspond to the edges of CH with length of $\beta_{max} = 2\pi - \frac{4\pi}{m}$. Since the number of points inside the major segments are less than r and also based on Lemma 3, all internal angles of the corresponding polygons are greater than or equal to $\frac{2\pi}{2^{r-1}m}$. Hence, all internal angles of the polygon computed by algorithm 2 are less than $2\pi - \frac{2\pi}{2^{r-1}m}$. □

In step 2.a of algorithm 2, for each edge of CH, the measure of sweeping arc expands from 0 to β_{max} and the sweeping arc contains the inner points as much as possible. In algorithm 3, presented below, the sweeping arcs that correspond to all edges of CH expand concurrently to contain all inner points. In this way, the upper bound is improved to $2\pi - \frac{2\pi}{2^{d-1}m}$ such that d is the depth of angular onion peeling on S which is defined as follows:

Let us increase the measure of all sweeping arcs concurrently from 0 to the first hit (or β_{max}, if a sweeping arc does not hit any point). All inner points that are hit by sweeping arcs form the layer 1 of points. The next layers are formed by deleting the points of the computed layer from inner points and keep increasing the measure of all sweeping arcs to the next hit. The process continues until all inner points are hit. The process of peeling away the layers, described above, is defined as *"angular onion peeling"* and the number of layers is called *"depth of angular onion peeling"* on these points.

Theorem 3. *There exists a polygon $P \in \wp(S)$ such that crosses S, and all internal angles of P are less than $2\pi - \frac{2\pi}{2^{d-1}m}$ where d denotes the depth of angular onion peeling on S.*

Proof. Here, by presenting algorithm 3, we construct such a polygon.

Algorithm 3

1. Compute CH as the convex hull of S and let IP be the set of inner points.
2. While IP is not empty:
   (a) Increase the measure of all sweeping arcs to the next hit or β_{max}.
   (b) Based on algorithm 1, reconstruct the polygons corresponding to each edge of CH.
   (c) Remove the visited points from IP.

All edges of corresponding polygons computed in step 2 minus all edges of CH except those that have no corresponding polygon, construct the desired polygon.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

Since, the number of points inside the major segments are less than d, all internal angles of corresponding polygons are greater than or equal to $\frac{2\pi}{2^{d-1}m}$. Hence, all internal angles of the polygon computed by algorithm 3 are less than $2\pi - \frac{2\pi}{2^{d-1}m}$. □

4.2. α-MAP vice versa α-MNP

Let S be a set of points on the grid G and $P \in \wp(S)$ be a simple polygon. Based on the Pick's theorem [50], the area of P is equal to $\frac{b}{2} + i - 1$ where b is the number of grid points on the boundary of P and i is the number of grid points which are inside the polygon P.

The polygon P crosses both b_1 points of S, which we call vertex points and b_2 non-vertex points on G, which we call grid points. Hence, $Area(P) = \frac{b_1+b_2}{2} + i - 1$.

Assume that polygons A and B with the same number of inner grid points cross no grid points, i.e. $b_2 = 0$. Hence, based on the Pick's theorem, the area of polygon A is more than that of B iff the number of vertex points in A is more than that in B. In this case, α-MAP is equivalent to α-MNP.

In the following, we show that for each polygon $P \in \wp(S)$ on the grid and all $\epsilon > 0$, there exists a polygon P' with the same vertices points such that $|Area(P) - Area(P')| < \epsilon$ and P' does not cross any grid point.

Let $e = \overline{ab}$ be an edge of P on the grid G. If $a = (x_a, y_a)$ and $b = (x_b, y_b)$, $W_e = |x_b - x_a|$ is the width of e and $H_e = |y_b - y_a|$ is the height of e. (see Fig. 16)

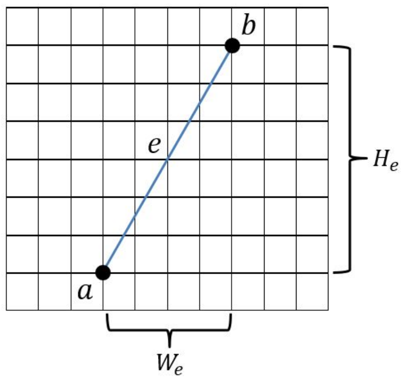

Figure 16. W_e and H_e are the width and height of e, respectively.

Lemma 4. *Let e be an edge of P on the grid G. If W_e and H_e are coprime integers, then e does not cross any grid point.*

Proof. Assume e crosses $n > 0$ grid points. As shown in Fig. 17, W_e is divided into $n + 1$ parts similar to H_e. Hence, $n + 1$ is common divisor of W_e and H_e. □

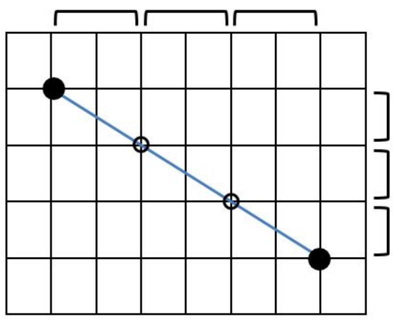

Figure 17. Common divisor of W_e and H_e is 3

 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

Lemma 5. *Let a and b be two non-coprime integers. There exist infinitely many positive integers $x > 1$ such that ax and $bx - 1$ are coprime integers.*

Proof. Each common denominator of a and b satisfies x. □

Definition 5. *The polygon $P \in \wp(S)$ is a grid avoiding polygon if P does not cross any grid points.*

The following theorem shows that if P crosses some grid points, for all $\epsilon > 0$ there exists a grid avoiding polygon P' such that $|Area(P) - Area(P')| < \epsilon$.

Theorem 4. *Let $e = \overline{ab}$ be an edge of $P \in \wp(S)$ that crosses a grid point. For all $\epsilon > 0$, there exists a point b' such that $e' = \overline{ab'}$ does not cross any grid point, the number of inner grid points does not change and $|Area(abb')| < \epsilon$.*

Proof. We convert the grid G to the grid G' by dividing each cell of G into x^2 subcells and place b' on the one grid point left or right of b, as shown in Fig. 18. If the right (left) grid point is inside the polygon P, place b' on the right (left) side of b. Let $n = xH_e$, $m = xW_e$ and $m' = xW_{e'} = m - 1$. Based on Lemma 5, there exist infinitely many integers x such that n and m' are coprime integers. Based on Lemma 4, since n and m' are coprime integers, the edge e' does not cross any grid point of G'. As seen in Fig. 18, the number of grid points inside the polygon does not change.

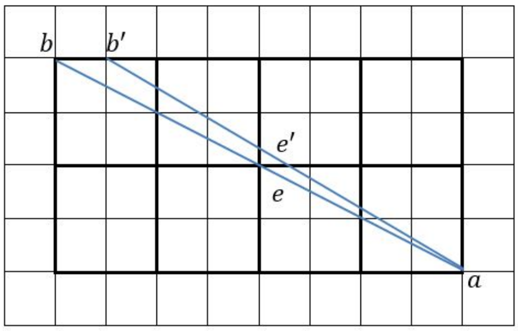

Figure 18. Grid G is shown in bold lines, and G' in regular mode

Let u be the length of each side of grid cells in G and u' be the length of each side of grid cells in G'. Based on Fig. 18, $Area(abb') = \frac{1}{2}u'H_e$. Since $u' = \frac{u}{x}$ and there exist infinitely many integers x such that $xW_{e'}$ and xH_e are coprime integers, for each $\epsilon > 0$ there exists an integer x such that $|Area(abb')| < \epsilon$. □

The following algorithm converts the grid G into the grid G' and the polygon P into the grid avoiding polygon P' on G'.

Algorithm 4

Let $P = (a_1, a_2, a_3, ..., a_n, a_1)$ be the polygon on the grid G.

1. Set $i = 1$
2. If $i = n$ go to 5, otherwise set $e = \overline{a_i a_{i+1}}$
3. If e does not cross any grid point
   (a) $i = i + 1$
   (b) Go to step 2
4. Else
   (a) Set $x = LCD(W_e, H_e)$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

  (b) Convert G into the grid G' using x. (dividing each cell of G into x^2 subcells)
  (c) For $j = 1$ to $i - 1$
    i. If $d = \overline{a_j a_{j+1}}$ crosses any grid point
      A. Move the vertex a_{j+1} to the left side or right side grid point (in G')
  (d) If e crosses any grid point
    i. Move the vertex a_{i+1} to the left side or right side grid point (in G')
  (e) set $i = i + 1$ and go to step 2
5. Exit

Let W_i be the width of $e_i = \overline{a_i a_{i+1}}$ and H_i be the length of e_i. Let us further assume that W_1 and W_2 are coprime to H_1 and H_2, respectively. Steps 3.a and 3.b avoid changing the position of these vertices. Assume $e_3 = \overline{a_3 a_4}$ crosses a grid point. The grid G is converted into the grid G' in step 4.b. In the new grid G', since $W_{1'}$=xW_1 and $H_{1'} = xH_1$, the width and length of e_1 are not coprime integers the same as the width and length of e_2. Therefore, the position of the previous vertices should be changed. Step 4.c of algorithm 4 updates the position of the previous vertices. Note that changing e_1 may have an effect on the edge e_2, hence, we check the loop in step 4.c to see if the edge crosses any grid point. If so, then, updating the last previous edge may have an effect on the edge e_3. Hence, in step 4.d, we change the position of a_4 if e_3 crosses any grid point. Finally, the position of all vertices are updated such that the new edges do not cross any grid point.

Corollary 1. *Let P_1 and P_2 be two simple polygons containing S with the same number of inner grid points. $Boundary(P_1) > Boundary(P_2)$ iff $Area(P_1) < Area(P_2)$. In other words, under these conditions we have the same solution for α-MAP and α-MNP.*

Corollary 2. *Let P_1 and P_2 be two simple polygons containing S. If $Boundary(P_1) \subset Boundary(P_2)$, then $Area(P_2) < Area(P_1)$.*

5. Numerical Experiments and Results

Considering equations (24), (28) and (31), the time complexity of the brute-force algorithm is $O(2^{n \cdot m \cdot r})$ such that n is the number of points, m is the number of vertices of CH and r is the number of inner points. In this section, we present a Genetic Algorithm as a fast and accurate method to solve these models. Genetic Algorithm is a powerful stochastic search technique that is applicable to a variety of nonconvex optimization problems [51].

In order to evaluate the results, we implemented both the GA and the brute-force algorithm for α-MAP, α-MPP and α-MNP. We ran both algorithms on the same datasets of points. Each dataset contained 100 sets of points with the same cardinality. We obtained the results for datasets of 5, 7, 10 and 12 points which are tabulated in Table 2.

A polygon-match occurs if the result of the GA on a set of points is the same as that of the brute-force algorithm. The quantity column in Table 2 shows the percentage of polygon-matches in each dataset and the quality column displays the average difference between the two areas, i.e. $Area(P) - Area(P')$ where P and P' are the constructed polygons using the genetic and the brute force algorithms, respectively.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 November 2018 doi:10.20944/preprints201811.0300.v1

Peer-reviewed version available at Mathematics 2018, 7, 24; doi:10.3390/math7010024

Table 2. Numerical results

number of points	number of generations	Area		Perimeter		Boundary	
		quality	quantity	quality	quantity	quality	quantity
n=5	50	99.15344	96	98.35035	95	98.6	95
	100	99.87179	99	99.46023	99	100	100
	150	100	100	100	100	100	100
	200	100	100	100	100	100	100
n=7	50	97.05038	83	97.1653	88	97.4026	82
	100	98.67008	92	98.77586	96	98.14285	93
	150	98.91394	93	100	100	100	100
	200	98.90656	95	100	100	100	100
n=10	50	94.21655	55	89.56722	67	80.90909	24
	100	95.14197	67	94.31243	80	89	50
	150	97.79823	83	96.67498	93	97.5	87
	200	97.88455	84	100	100	100	100
n=12	50	87.95903	52	87.86893	59	80.53846	20
	100	90.84784	63	93.45262	70	84.66667	44
	150	96.67498	77	95.95328	84	93.91667	73
	200	97.76408	82	96.15958	98	98.5	91

The pseudo code for the GA is presented as follows:

Algorithm 5 (Genetic Algorithm)

Inputs:

S : the points set α : constraint on angles
e : elitism rate mu : mutation rate
itt_{count} : iteration count

1. Initialize the population: generate random α-polygons with $m, m+1, ..., n$ vertices containing S. Also, set $itt = 1$.
2. Coding: Compute the vector C for each randomly generated polygon as a code such that $C[i \times j \times k] = 1$ iff $Z_{i,j,k}$=1. The length of C is $n \times m \times r$.
3. Packing: Construct a chromosome chr for each code such that $chr[k] = j$ iff the kth inner point is assigned to the jth edge of CH. The length of chr is r.
4. Elitist selection: Select e percent of the best chromosomes and move them to the next generation.
5. Crossover: The single-point crossover is used. Select a random position ind $(1 < ind < r)$ and two random chromosomes as the parents.
6. Mutation: Each child that is constructed in step 5 is mutated with a probability of mu. Select a random position ind $(1 < ind < r)$ and randomly change the value of $chr[ind]$. The mutation leads an inner point to be assigned to another edge.
7. Unpacking: Convert each chromosome of the new generation into an individual code. Each chr in the new generation is unpacking to a code C. In this step, the order of assigned points for each edge is specified.
8. Re-evaluate: Based on the objective function (Area, Perimeter and Boundary), re-evaluate the new polygons and keep the best chromosome as the solution. Replace the old generation with the new one and set $itt = itt + 1$. If $itt < itt_{count}$ go to step 3, otherwise finish.

Because of the exponential time complexity of the brute-force algorithm, the exact result could not be obtained on large datasets in a reasonable computational time. Thus, we ran the GA on datasets of 15, 20, 25 and 30 points and displayed the resulting polygons in Fig. 19 which are the solutions for α-MAP for $\alpha = \pi$.

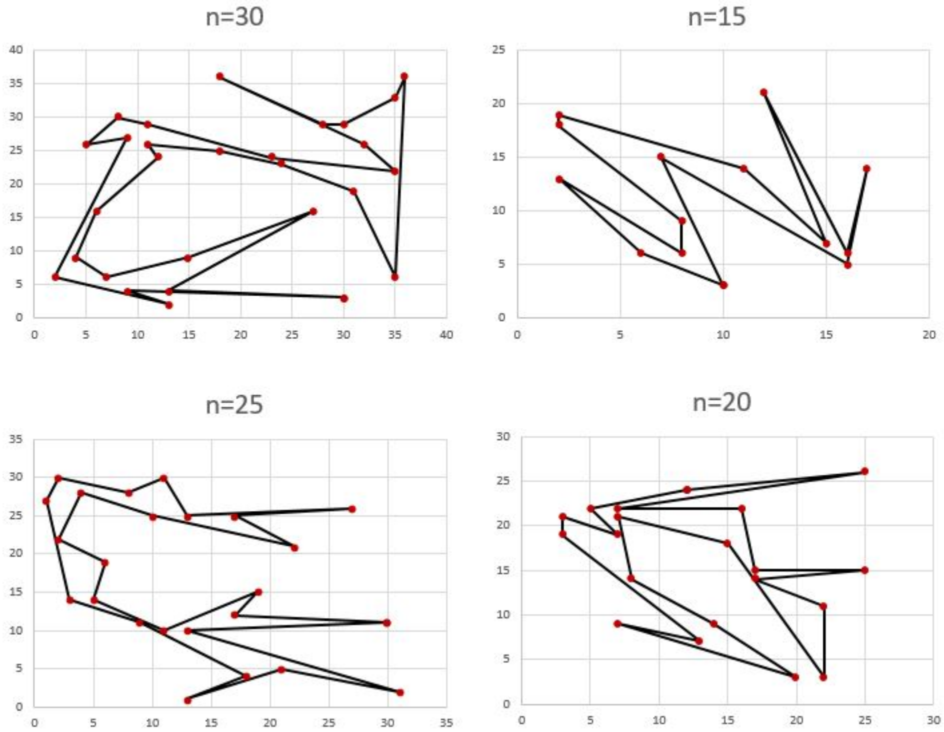

Figure 19. Solutions for α-MAP for $\alpha = \pi$ and n =15, 20, 25 and 30

As stated in section 1, concave hull is a generalization of convex hull that identifies the area occupied by a set of points. Moreira and Santos presented an algorithm to compute concave hull [47], and in [52] an algorithm was presented to compute concave hull in d dimensions. We implemented the said algorithm in [47] and compared the quality of the obtained result with that of the GA. Fig. 20 illustrates this comparison with the x-axis exhibiting the cardinality of datasets and the y-axis exhibiting the approximation average quality of the results.

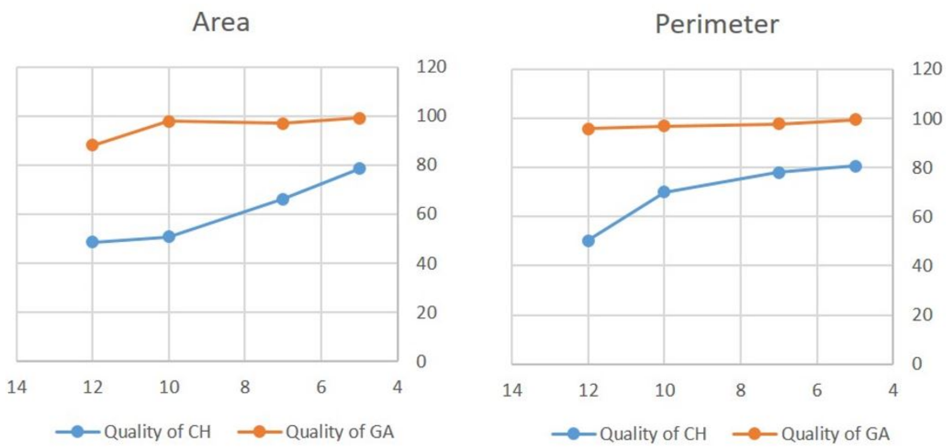

Figure 20. GA versus Concave Hull algorithm (CH) in terms of both area and perimeter metrics

Figs. 21, 22 and 23 illustrate the results of GA on a set of points which are the solutions for α-MAP, α-MPP and α-MNP for different values of α, respectively. For $\alpha = 0$, the constructed polygons are the convex hull of points. As the value of α increases, the concave angles start to appear in the polygons. For the large values of α, the boundary of polygons would pass through all of the points, e.g. for $\alpha = 180°$, the results are simple polygons with approximately minimum area, maximum perimeter and maximum number of vertices, respectively.

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

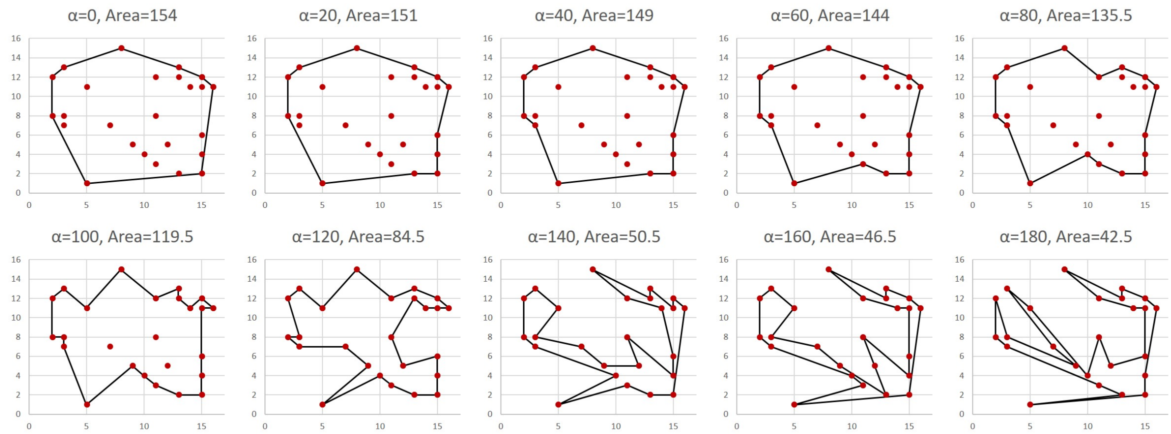

Figure 21. Application of GA to approximate α-MAP for different values of α

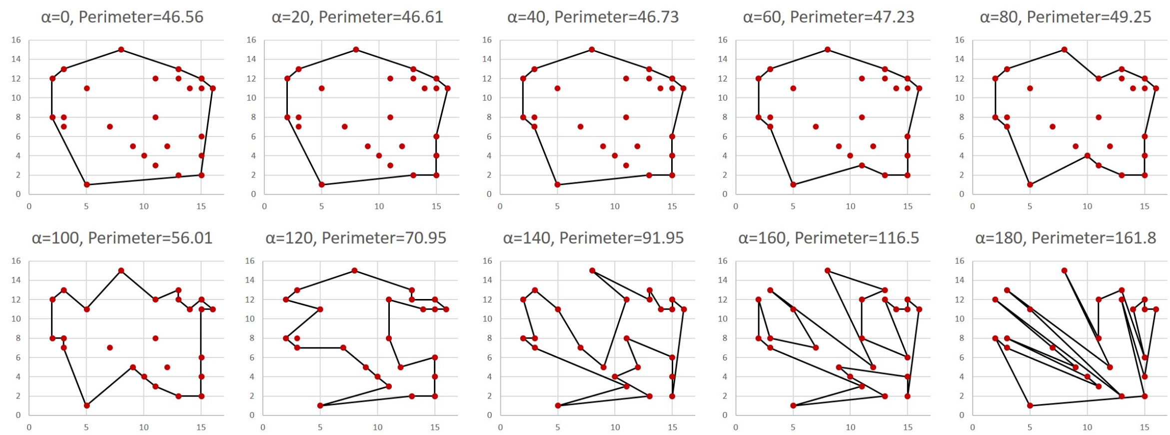

Figure 22. Application of GA to approximate α-MPP for different values of α

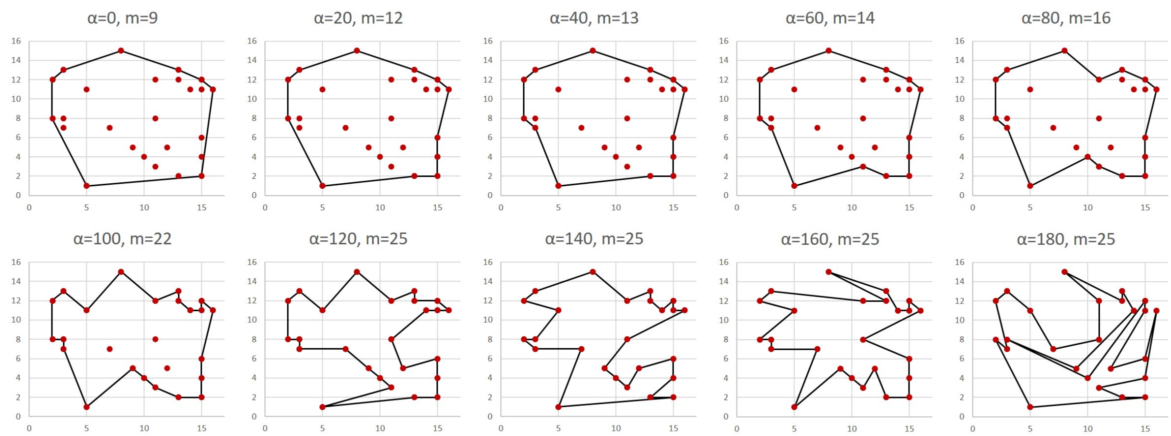

Figure 23. Application of GA to approximate α-MNP for different values of α

6. Conclusion

In this paper, we considered the problem of finding optimal simple polygons containing a set of points in the plane. We generalized the problems of finding the minimum area, maximum perimeter and maximum number of vertices containing a set of points by adding constraint to the angles of polygons. We formulated the generalized problems as nonlinear programming models.

Peer-reviewed version available at *Mathematics* **2018**, *7*, 24; doi:10.3390/math7010024

Given that all simple polygons contain a set of points, we derived an upper bound for the minimum of the maximum angles of each polygon. As a further theoretical achievement, we demonstrated that the problem of computing polygon with the minimum area is almost equivalent to that of computing polygon with the maximum number of vertices.

We presented a genetic algorithm to solve these models and conducted experiments on several datasets. At the end, in comparison to the brute-force method and other previous studies, better results were obtained.

References

1. Papadimitriou, C.H. The Euclidean travelling salesman problem is NP-complete. *Theoretical computer science* **1977**, *4*, 237–244.
2. Fekete, S.P.; Pulleyblank, W.R. Area optimization of simple polygons. Proceedings of the ninth annual symposium on Computational geometry. ACM, 1993, pp. 173–182.
3. Pakhira, M.K. *Digital image processing and pattern recognition*; PHI Learning Pvt. Limited, 2011.
4. Marchand-Maillet, S.; Sharaiha, Y.M. *Binary digital image processing: a discrete approach*; Academic Press, 1999.
5. Pavlidis, T. *Structural pattern recognition*; Vol. 1, Springer, 2013.
6. Abdi, M.N.; Khemakhem, M.; Ben-Abdallah, H. An effective combination of MPP contour-based features for off-line text-independent arabic writer identification. In *Signal processing, image processing and pattern recognition*; Springer, 2009; pp. 209–220.
7. Galton, A.; Duckham, M. What is the region occupied by a set of points? *Geographic Information Science* **2006**, pp. 81–98.
8. Li, X.; Frey, H.; Santoro, N.; Stojmenovic, I. Strictly localized sensor self-deployment for optimal focused coverage. *IEEE Transactions on Mobile Computing* **2011**, *10*, 1520–1533.
9. Nguyen, P.L.; Nguyen, K.V. Hole Approximation-Dissemination Scheme for Bounded-Stretch Routing in Sensor Networks. Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on. IEEE, 2014, pp. 249–256.
10. Lawler, E.L.; Lenstra, J.K.; Kan, A.R.; Shmoys, D.B.; others. *The traveling salesman problem: a guided tour of combinatorial optimization*; Vol. 3, Wiley New York, 1985.
11. Asaeedi, S.; Didehvar, F.; Mohades, A. α-Concave hull, a generalization of convex hull. *Theoretical Computer Science* **2017**, *702*, 48–59.
12. Fekete, S.P.; Woeginger, G.J. Angle-restricted tours in the plane. *Computational Geometry* **1997**, *8*, 195–218.
13. Culberson, J.; Rawlins, G. Turtlegons: generating simple polygons for sequences of angles. Proceedings of the first annual symposium on Computational geometry. ACM, 1985, pp. 305–310.
14. Evans, W.S.; Fleszar, K.; Kindermann, P.; Saeedi, N.; Shin, C.S.; Wolff, A. Minimum Rectilinear Polygons for Given Angle Sequences. In *Discrete and Computational Geometry and Graphs*; Springer, 2015; pp. 105–119.
15. Cho, H.G.; Evans, W.; Saeedi, N.; Shin, C.S. Covering points with convex sets of minimum size. International Workshop on Algorithms and Computation. Springer, 2016, pp. 166–178.
16. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* **1960**, *7*, 326–329.
17. Fasano, G. A global optimization point of view to handle non-standard object packing problems. *Journal of Global Optimization* **2013**, *55*, 279–299.
18. Liu, H.; Liu, W.; Latecki, L.J. Convex shape decomposition. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 97–104.
19. Masehian, E.; Habibi, G. Robot path planning in 3D space using binary integer programming. *International Journal of Mechanical, Industrial and Aerospace Engineering* **2007**, *1*, 26–31.
20. Kallrath, J. Cutting circles and polygons from area-minimizing rectangles. *Journal of Global Optimization* **2009**, *43*, 299–328.
21. Speckmann, B.; Kreveld, M.; Florisson, S. A linear programming approach to rectangular cartograms. *Progress in Spatial Data Handling* **2006**, pp. 529–546.
22. Seidel, R. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry* **1991**, *6*, 423–434.

23. Peethambaran, J.; Parakkat, A.D.; Muthuganapathy, R. An Empirical Study on Randomized Optimal Area Polygonization of Planar Point Sets. *Journal of Experimental Algorithmics (JEA)* **2016**, *21*, 1–10.
24. Taranilla, M.T.; Gagliardi, E.O.; Hernández Peñalver, G. Approaching minimum area polygonization **2011**.
25. Moylett, D.J.; Linden, N.; Montanaro, A. Quantum speedup of the traveling-salesman problem for bounded-degree graphs. *Physical Review A* **2017**, *95*, 032323.
26. Bartal, Y.; Gottlieb, L.A.; Krauthgamer, R. The traveling salesman problem: low-dimensionality implies a polynomial time approximation scheme. *SIAM Journal on Computing* **2016**, *45*, 1563–1581.
27. Hassin, R.; Rubinstein, S. Better approximations for max TSP. *Information Processing Letters* **2000**, *75*, 181–186.
28. Dudycz, S.; Marcinkowski, J.; Paluch, K.; Rybicki, B. A 4/5-Approximation Algorithm for the Maximum Traveling Salesman Problem. International Conference on Integer Programming and Combinatorial Optimization. Springer, 2017, pp. 173–185.
29. Matei, O.; Pop, P. An efficient genetic algorithm for solving the generalized traveling salesman problem. Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on. IEEE, 2010, pp. 87–92.
30. Lin, B.L.; Sun, X.; Salous, S. Solving travelling salesman problem with an improved hybrid genetic algorithm. *Journal of computer and communications.* **2016**, *4*, 98–106.
31. Hussain, A.; Muhammad, Y.S.; Nauman Sajid, M.; Hussain, I.; Mohamd Shoukry, A.; Gani, S. Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. *Computational Intelligence and Neuroscience* **2017**, *2017*.
32. Jakobs, S. On genetic algorithms for the packing of polygons. *European journal of operational research* **1996**, *88*, 165–181.
33. Parvez, W.; Dhar, S. Path planning of robot in static environment using genetic algorithm (GA) technique. *International Journal of Advances in Engineering & Technology* **2013**, *6*, 1205.
34. Vadakkepat, P.; Tan, K.C.; Ming-Liang, W. Evolutionary artificial potential fields and their application in real time robot path planning. Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. IEEE, 2000, Vol. 1, pp. 256–263.
35. Dalai, J.; Hasan, S.Z.; Sarkar, B.; Mukherjee, D. Adaptive operator switching and solution space probability structure based genetic algorithm for information retrieval through pattern recognition. Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on. IEEE, 2014, pp. 1624–1629.
36. Duckham, M.; Kulik, L.; Worboys, M.; Galton, A. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition* **2008**, *41*, 3224–3236.
37. Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on* **1983**, *29*, 551–559.
38. Amenta, N.; Bern, M.; Eppstein, D. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing* **1998**, *60*, 125–135.
39. Ganapathy, H.; Ramu, P.; Muthuganapathy, R. Alpha shape based design space decomposition for island failure regions in reliability based design. *Structural and Multidisciplinary Optimization* **2015**, *52*, 121–136.
40. Fayed, M.; Mouftah, H.T. Localised alpha-shape computations for boundary recognition in sensor networks. *Ad Hoc Networks* **2009**, *7*, 1259–1269.
41. Ryu, J.; Kim, D.S. Protein structure optimization by side-chain positioning via beta-complex. *Journal of Global Optimization* **2013**, *57*, 217–250.
42. Varytimidis, C.; Rapantzikos, K.; Avrithis, Y.; Kollias, S. α-shapes for local feature detection. *Pattern Recognition* **2016**, *50*, 56–73.
43. Siriba, D.N.; Matara, S.M.; Musyoka, S.M. Improvement of Volume Estimation of Stockpile of Earthworks Using a Concave Hull-Footprint. *International Scientific Journal for Micro, Macro and Mezzo Geoinformation* **2015**, *5*.
44. Chau, A.L.; Li, X.; Yu, W. Large data sets classification using convex–concave hull and support vector machine. *Soft Computing* **2013**, *17*, 793–804.
45. Vishwanath, A.; Ramanathan, M. Concave hull of a set of freeform closed surfaces in R 3. *Computer-Aided Design and Applications* **2012**, *9*, 857–868.
46. Jones, J. Multi-agent Slime Mould Computing: Mechanisms, Applications and Advances. In *Advances in Physarum Machines*; Springer, 2016; pp. 423–463.

47. Moreira, A.; Santos, M.Y. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points **2007**.
48. Braden, B. The surveyor's area formula. *The College Mathematics Journal* **1986**, *17*, 326–337.
49. Graham, R.L. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters* **1972**, *1*, 132–133.
50. Pick, G. Geometrisches zur zahlenlehre. *Sitzenber. Lotos (Prague)* **1899**, *19*, 311–319.
51. Qing-feng, Z. The Application of Genetic Algorithm in Optimization Problems. *Journal of Shanxi Normal University (Natural Science Edition)* **2014**, *1*, 008.
52. Park, J.S.; Oh, S.J. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of information science and engineering* **2013**, *29*, 379–392.