Two Efficient approaches to solving Robot Arm's folding problem in $\mathbb{R}^2$

(Can we find an efficient algorithm to solve Robot Arm's folding problem in $\mathbb{R}^2$?)

Word Count: 3990

Table of Contents

1. Introduction

Folding and unfolding problems have been increasingly studied in modern mathematics. In general, they can be partitioned into three branches, corresponding to three dimensions, where linkages (1D), paper (2D) and polyhedra (3D) are focused on. While the ultimate issue of this kind is to determine the "foldability" of a given material, lots of theorems and folding algorithms can be developed and applied to computational and engineering problems in the real world. Furthermore, they can even be mapped to other NP-complete problems so that elegant relationships are revealed.[1]

In this paper, we particularly focus on a commonly used "linkage" material, robot arm, and investigate its folding properties. More specifically, a robot arm is made up of n arm segments, where each two adjacent ones are connected by hinges. In mathematics, it's defined as $A_n = (l_1, l_2 \dots l_n)$, where l_i is the length of each arm segment. [2]

Based on these prerequisites, our ultimate goal is to see given a point P in a robot arm's reachability region, how we can find a configuration of the arm that places the hand on P. Especially, in today's industrialized world, more flexible and efficient robot arms are demanded and implemented in people's daily life; and as a programmer of our school's robot team, I hold the opinion that these questions behind should not only be solved in an engineering way,

---

[1] O'Rourke, Joseph. "*How to fold it.*" Cambridge University Press, 2007.

[2] O'Rourke, Joseph. Demaine, Erik D. *"Geometric Folding Algorithms: Linkages, Origami, Polyhedra."* Cambridge University Press, 2007. (p.p. 1~16)

but also need robust algorithms to boost the arm's speed and accuracy. Therefore, we're motivated to tackle its mathematical issues in our paper.

2. Robot Arms

2.1 Definition

Nowadays, robot arms are prevailingly applied in different fields such as automated processing and space exploitation. To describe, an robot arm is made up of n segments (where $n > 0$ and $\in \mathbb{Z}^+$). The first arm segment is connected to a fix point (i.e. a defined origin 0), and each subsequent arm segment is connected to the previous one at the endpoint by hinges. Each arm segment can rotate however it wants to fulfill their jobs (reach a target, fold together, etc.), but they're not allowed to cross over each other.

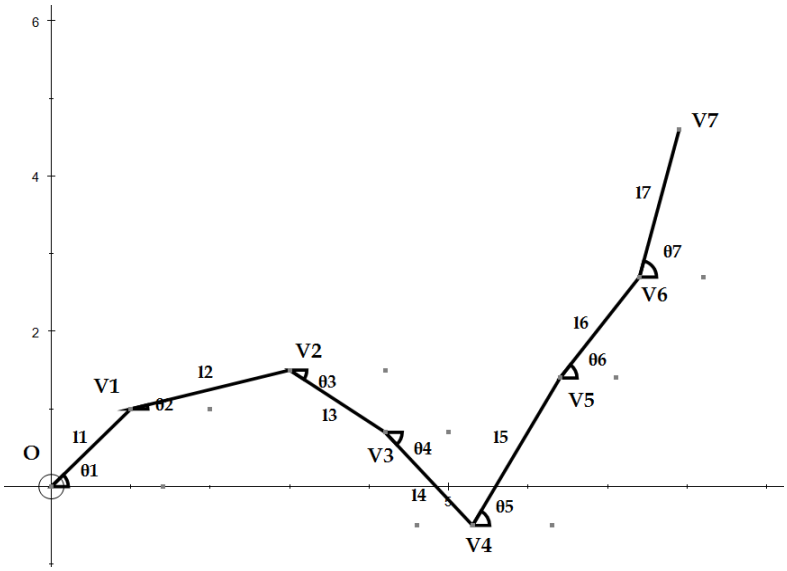

Fig 1. A robot arm with 7 segments

In this paper, we particularly focus on a robot arm in $\mathbb{R}^2$ with Cartesian/Polar coordinates. Mathematically, we define a robot arm in terms of its segments as $A_n = (l_1, l_2 \dots l_n)$ where l_i is the length of the i^{th} segment (Figure 1.).[3]

[3] A similar definition is given in: O'Rourke, Joseph. "How to fold it." Cambridge University Press, 2007. (p.p. 3~4)

2.2 Configuration

Given a robot arm defined by its arm segments, the configuration of the arm is determined by how each segment is bended (rotates) at its beginning point (around the hinges). Mathematically, we define the configuration of a robot arm $A_n = (l_1, l_2 \dots l_n)$ as $C_n = (\theta_1, \theta_2 \dots \theta_n)$, where θ_i is the beginning bend (rotation) of the i^{th} arm segment. $(-\pi \leq \theta_i \leq \pi)$ (Figure 1.)

3. Reachability Region

To investigate "how" a robot arm can reach a given (target) point, we first have to determine "where" an arm can reach by its length and configuration so that we can give a reasonable inquiry.[4]

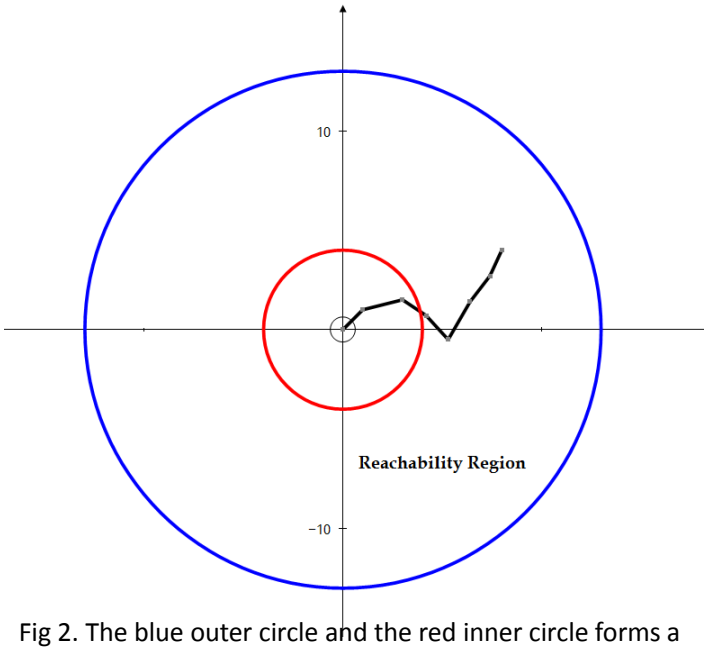

Fig 2. The blue outer circle and the red inner circle forms a robot arm's reachability region (i.e. an annulus)

By the definition, as the robot arm moves (rotates) around a fixed point,

[4] A more detailed definition is given in: O'Rourke, Joseph. "How to fold it." Cambridge University Press, 2007. (p.p. 5)

namely "center", we intuitively conjectured that a robot arm's reachability region is a circle or annulus just as it shows in Figure 2. However, to confirm the accurate shape of a robot arm's reachability region, the radius of such an annulus has to be proved mathematically.

Claim 1. *Given a robot arm* $A_n \in \mathbb{R}^2$, *the arm's reachability region is an annulus with an outer radius* $R = \sum_{i=1}^{n} l_i$ *and an inner radius* $r = 2M - \sum_{i=1}^{n} l_i$, *where* $M = max_{1 \le i \le n}\, l_i$ *(If negative, then* $r = 0$*).*

Proof:

Lemma 1. *The end point (*v_n *or hand) of a robot arm* A_n *can be described by the vector addition of all its segments (Figure 3.)*

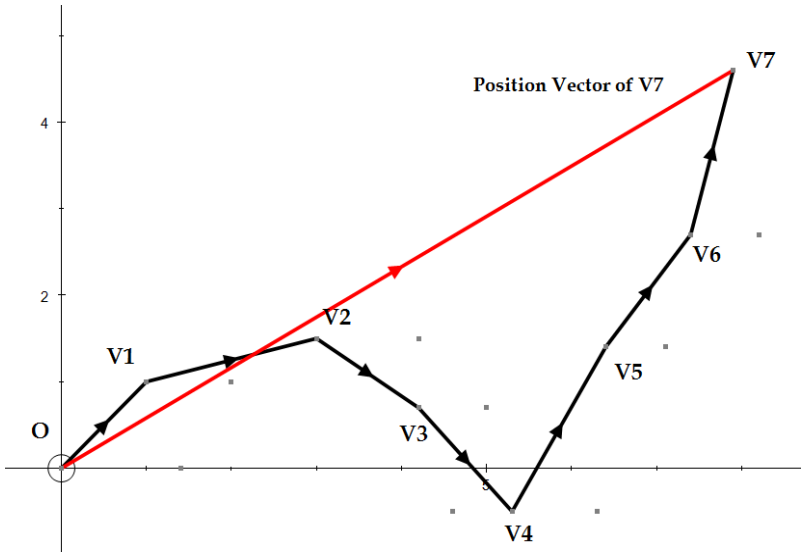

Fig 3.The position vector $\overrightarrow{v_7}$ is the vector addition of all segment vectors

Proof for Lemma 1. For the sake of simplicity, consider a robot arm centers at the origin of $\mathbb{R}^2$, then the end point v_n can be described in position vector form $\overrightarrow{v_n}$. Now consider all the segments of A_n as vectors $\overrightarrow{l_1}, \overrightarrow{l_2} \ldots \overrightarrow{l_n}$ in $\mathbb{R}^2$. By

the Parallelogram law[5], the geometric implication of all these vectors' addition is perfectly equals the position vector $\overrightarrow{v_n}$, i. e. $\overrightarrow{l_1} + \overrightarrow{l_2} + \cdots + \overrightarrow{l_n} = \overrightarrow{v_n}$. Hence, **lemma 1**. is valid for any robot arm A_n.

Q. E. D.

Based on **Lemma 1.**, by the commutativity of vector addition, we deduced the following lemma:

Lemma 2. *A robot arm's reachability region is independent of the order of segments.*

In other words, given the same segments to build a robot arm, its capacity/reachability region is the same regardless how the segments are connected. Accordingly, we proved the primal statement as it follows:

Proof for Claim 1. Without loss of generality, by **lemma two**, assume l_1 is the largest segment for a robot arm $A_n = (l_1, l_2 \dots l_n)$ centering at a defined origin 0, then the reachability region is the trajectory of its end point v_n.

Consider v_n can reach an arbitrary point A in $\mathbb{R}^2$, where $0A = |\overrightarrow{v_n}|$. By rotating the robot arm, $0A$ is rotated around the center 0 and the trajectory of $0A$ is a circle. Thus, the range of $0A$ for a robot arm determines its reachability region.

For the maximum value of $0A$, since $|\overrightarrow{v_n}| = \left| \overrightarrow{l_1} + \overrightarrow{l_2} + \cdots + \overrightarrow{l_n} \right|$ is maximum when the linear interpolation of $v_1, v_2 \dots v_n$is a straight line, clearly it will be found when the robot arm is straightened, i. e. $C_n = (0,0, \dots 0)$. In this case, $0A$ is the sum of all the arm segments, i. e. $l_1 + l_2 + \cdots + l_n$. This is the furthest

[5] Weisstein, Eric W. "Polar Coordinates." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/PolarCoordinates.html

away we can get as it shows in Figure 4.

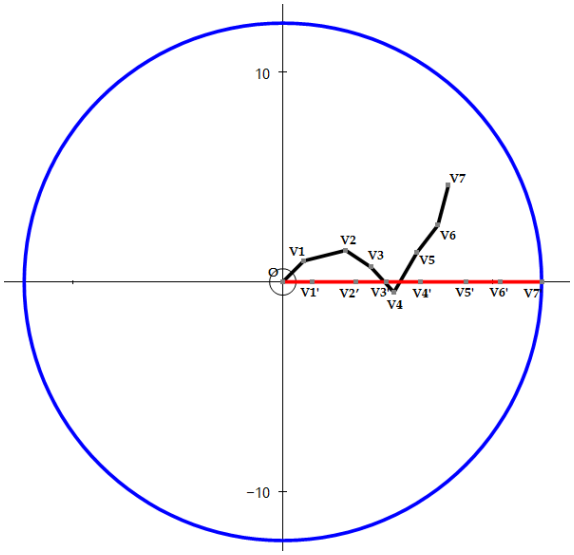

Fig 5. The robot arm folds like a closed polygon (the red line is the longest segment)

For the minimum value of OA, we want to minimize $\left|\overrightarrow{l_1}+\overrightarrow{l_2}+\cdots+\overrightarrow{l_n}\right|$.

Case one. If $l_1 < l_2+\cdots+l_n$, then we can make a polygon out of the vectors by folding $l_2, \ldots l_n$ like Figure 5. Hence, we can make $\overrightarrow{v_n}=\overrightarrow{l_1}+\overrightarrow{l_2}+\cdots+\overrightarrow{l_n}=\vec{0}$, i. e. $OA = 0$. Namely, the minimum value of OA is 0 here.

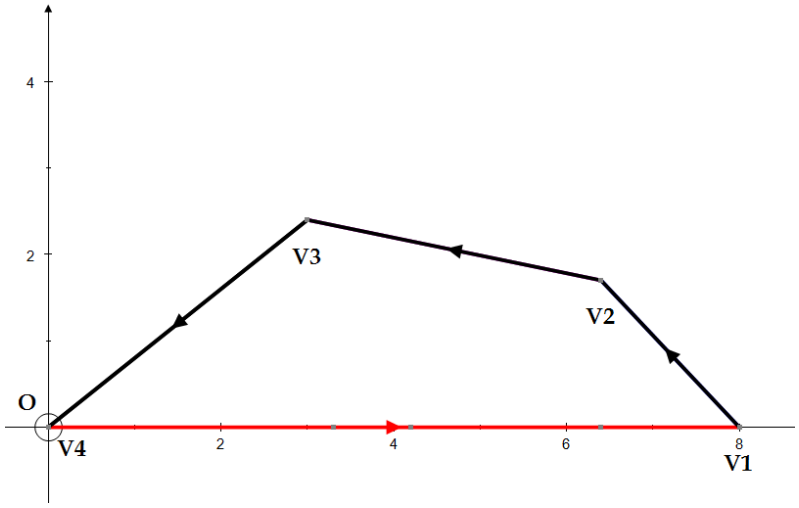

Fig 4.The outer border (blue circle) of a robot arm's R.R. and a straightened robot arm (red line)

Case two. If $l_1 \geq l_2 + \cdots + l_n$, then by the triangle inequality: for vectors $\vec{x}$, $\vec{y}$, $|\vec{x} + \vec{y}| \leq |\vec{x}| + |\vec{y}|$,[6]

$$|\overrightarrow{v_n}| = |\overrightarrow{l_1} + \overrightarrow{l_2} + \cdots + \overrightarrow{l_n}| \geq |\overrightarrow{l_1}| - |\overrightarrow{l_2} + \cdots + \overrightarrow{l_n}|$$

$$\geq |\overrightarrow{l_1}| - (|\overrightarrow{l_2}| + \cdots + |\overrightarrow{l_n}|) = l_1 - (l_2 + \cdots + l_n)$$

Namely, the minimum value of OA is $l_1 - (l_2 + \cdots + l_n)$ in this case. It can be achieved if $l_2, \ldots, l_n$ are opposite l_1 geometrically.

Combining case one and two, the minimum value of OA is $l_1 - (l_2 + \cdots + l_n)$ (If negative, $\min OA = 0$).

Now starting from a configuration of A_n with the minimum length (i. e. OA), straighten $v_2, v_3 \ldots v_n$ orderly so that the bend of at each hinge becomes 0.

Then, OA can take any value between its minimum and maximum. Thus, the range of OA is $l_1 - (l_2 + \cdots + l_n) \leq OA \leq l_1 + l_2 + \cdots + l_n$, where OA is non-negative (Figure 6.).

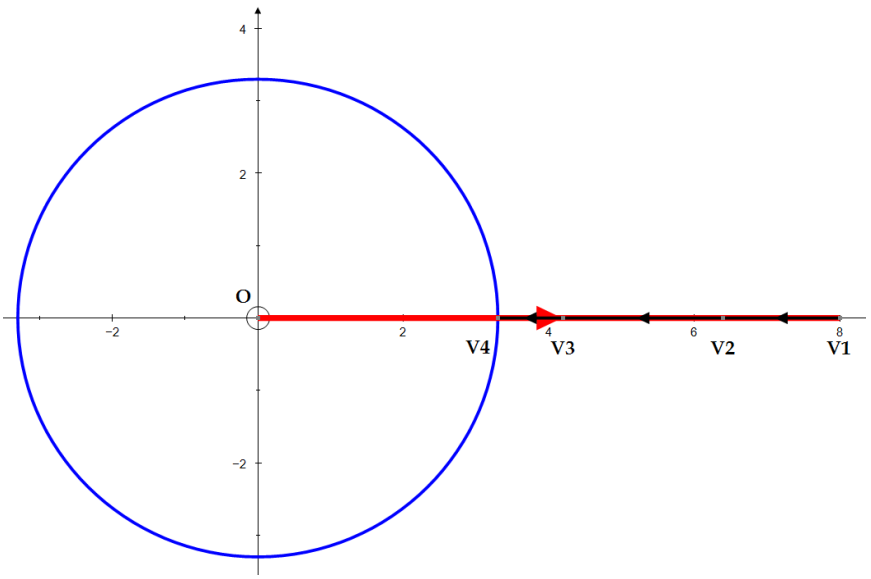

Fig 6.The inner border (Blue circle) of the robot arm's R.R. The longest segment (red line) folds opposite the remaining ones

[6] Weisstein, Eric W. "Triangle Inequality." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/TriangleInequality.html

Hence, according to what we discussed about the shape of a robot arm's reachability region, it's an annulus with an outer radius $\mathrm{R} = \sum_{i=1}^{n} l_i$ and inner radius $\mathrm{r} = 2\mathrm{M} - \sum_{i=1}^{n} l_i$, where $\mathrm{M} = \max_{1 \le i \le n} l_i$ (If negative, then $\mathrm{r} = 0$).

Q. **E**. **D**.

4. Robot Arms' folding problem

Knowing the reachability region of a robot arm $A_n \in \mathbb{R}^2$, in this section, we will particularly take an insight into efficient approaches to finding a configuration of A_n given a point P in its reachability. To start with, we'll first explore some general ideas about the essence of our solutions. Based on these discoveries, we'll then develop two approaches, both algebraically and geometrically, and compare in the next section. For each approach, we'll have an approach outline and a detailed description. Some neat proofs will also be given after the solution if applicable.

4.1 Problem Definition

Given a point P in a robot arm A_n's reachability region, what angles at $v_0, v_1, \ldots v_{n-1}$ (i. e. the configuration of A_n) will make its hand v_n reaches P (Figure 7.).[7]

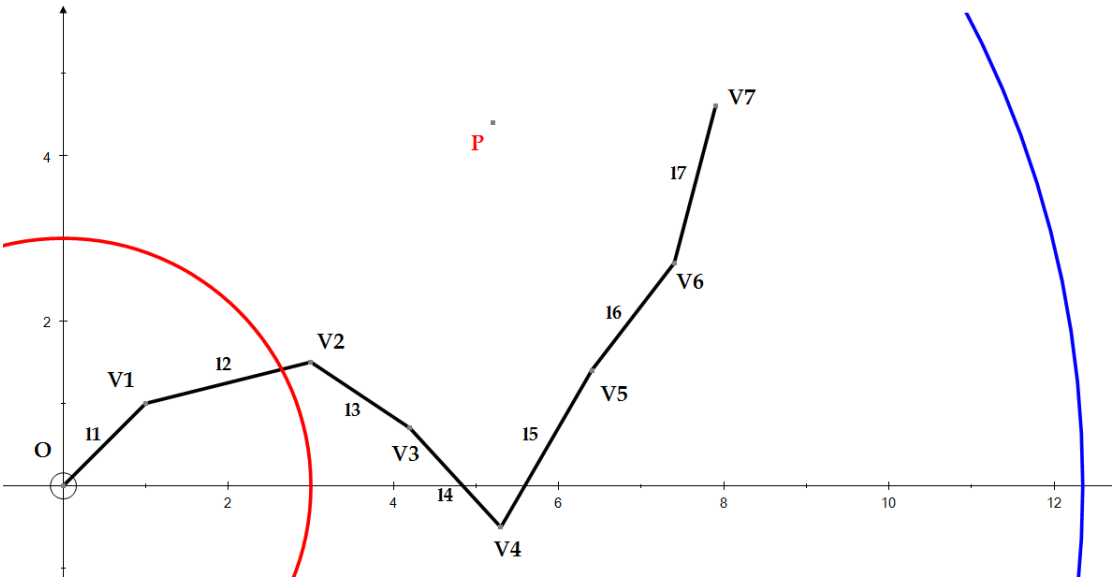

Fig 7. Robot Arm's folding problem (red circle: inner border of R.R.; blue circle: outer border of R.R.)

[7] A similar definition is given in: O'Rourke, Joseph. "How to fold it." Cambridge University Press, 2007. (p.p. 18)

4.2 Basic Strategy

In the proof of robot arm's reachability region, we deduced the lemma that the end point (v_n or hand) of a robot arm A_n can be described by the vector addition of all its segments. This gives us an opening to think about the folding problem computationally by coping with the arm segments using vectors. Accordingly, in $\mathbb{R}^2$, each arm segment l_i can be expressed in the polar form $A_i \cdot cis\theta_i$[8], where A_i is the magnitude of l_i and θ_i is the bending angle at v_{i-1}. Hence, algebraically, we can work out a robot arm's configuration by solving a system of equations that fulfills our goal.

Geometrically, as an end point (target) P and the last arm segment l_n are both known in our problem, we may be able to start from the last point and confirm each point backwards by exhausting all their possible positions. Specifically, we may take the last arm segment l_i as the radius and its end point v_{i+1} as the center so that the circle is the trajectory of its beginning point v_i, and iterate. Similarly to the algebraic method, we can also describe each hinge's coordinates with trigonometry for computation

[8] Weisstein, Eric W. "Polar Coordinates." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/PolarCoordinates.html

4.3 Approach A: Polar Method

4.3.1 Approach Outline

In this method, we'd like to manipulate the arm segments in vector forms in a way that equations of can be derived and solved. To express such manipulation, instead of Cartesian-form coordinates of each vertex (hinge), Polar-form coordinates are preferred so that the direction of each segment can be manipulated (Figure 8.).

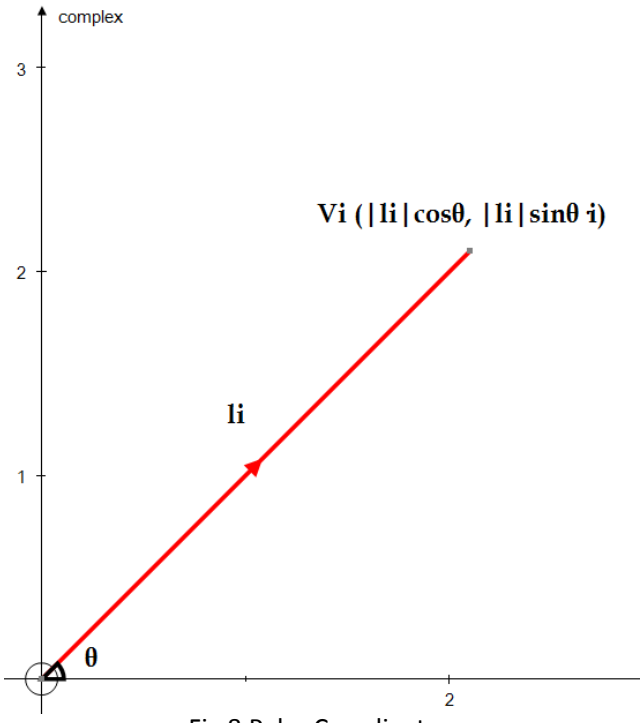

Fig 8.Polar Coordinates

Based on that, as the end point (v_n or hand) of a robot arm A_n can be described by the vector addition of all its segments, we can easily derive an equation with the sum of all the segment vectors in polar-form coordinates as the left-hand side and the position vector of P in polar-form coordinates as the right-hand side. Finally, while solving the equation, sorting the real part and complex part of the equation, a system of two equations will be obtained.

4.3.2 Calculation

For the sake of simplicity, assume the robot arm A_n centers at the origin of polar coordinates. Then the position vector of target point P can be expressed as $P = a + b \cdot i$, where $a, b \in \mathbb{R}$.

Now consider the first segment of A_n, l_1, with a magnitude of $|l_1|$ and a bending angle of θ_1. Then the polar-form coordinates of $l_1 = |l_1| cis\theta_1$. Similarly, the rest of the arm segments are $l_2 = |l_2| cis\theta_2 \ldots l_n = |l_n| cis\theta_{n-1}$. Since the end point ($v_n$ or hand) of a robot arm A_n can be described by the vector addition of all its segments, we reach the following equation:

$$|l_1| cis\theta_1 + \cdots + |l_n| cis\theta_{n-1} = a + b \cdot i$$

where the magnitude of the arm segments $|l_i|$, a, b are all known.

Sorting the real part and complex part of the equation, a system of two equations are derived below:

$$\begin{cases} \sum_{i=1}^{n} |l_i| \cdot \cos\theta_i = a \\ \sum_{i=1}^{n} |l_i| \cdot sin\theta_i = b \end{cases}$$

This is a system of linear equation with n unknowns. Equivalently, it's a $n \times 2$ constant matrix times a $1 \times n$ unknown matrix equals a 2×1 matrix. Thus, as we're certain that the target point P is reachable (which means the result cannot have no solution), the result will have $(n - 2)$-dimensional space of solutions.

4.3.3 Approach Evaluation

For evaluation, we'd consider both the feasibility for computing and efficiency of the approach. First, the algebraic bash is practical, intuitive and simple. The whole process is algebraic so that it's easy to be transformed into computer language. Also, it'd be robust if the amount of robot arm segments is limited (e.g. 2 to 4), which is close to real-life situations.

However, when the amount of arm segments largely increases for space or other special uses, the calculation will be far beyond human capacity. In fact, even for computers, since they have to exhaust all the possible solutions for the system of linear equation, when the amount of arm segments largely increases, the computational complexity will increase enormously as well. This increase can be far from the polynomial time and makes the problem unsolvable.

4.4 Approach B: Geometric Method

4.4.1 Approach Outline

To visualize and improve our method, a smarter geometric method is developed here using the Cartesian coordinates. As we mentioned before, since the end point, v_n, and the whole arm segment are given, it gives us an opening to start constructing the arm backwards and finally, to the origin. Specifically, considering v_n as the center and the last segment l_n as the radius, the resultant circle is to be the trajectory of all the possible positions of v_{b-1}. This leads us to a discussion of two cases.

Case 1. If the circle n intersects with the border of the remaining (n − 1)-segment arm's reachability region; then take one intersection as the position of hinge v_{n-1}, since the remaining segments can reach the intersection by straightened, folding opposite the longest segment, or forming a closed polygon, we can output one entire robot arm's configuration. A diagram for visualization is provided below (Figure 9.). Notably, it's when the "hinge circle" intersects with the outer border of the remaining arm's reachability region. When it intersects with the inner border, a same graph can be plotted except the hinge is on the intersection between the "hinge circle" and the inner border.

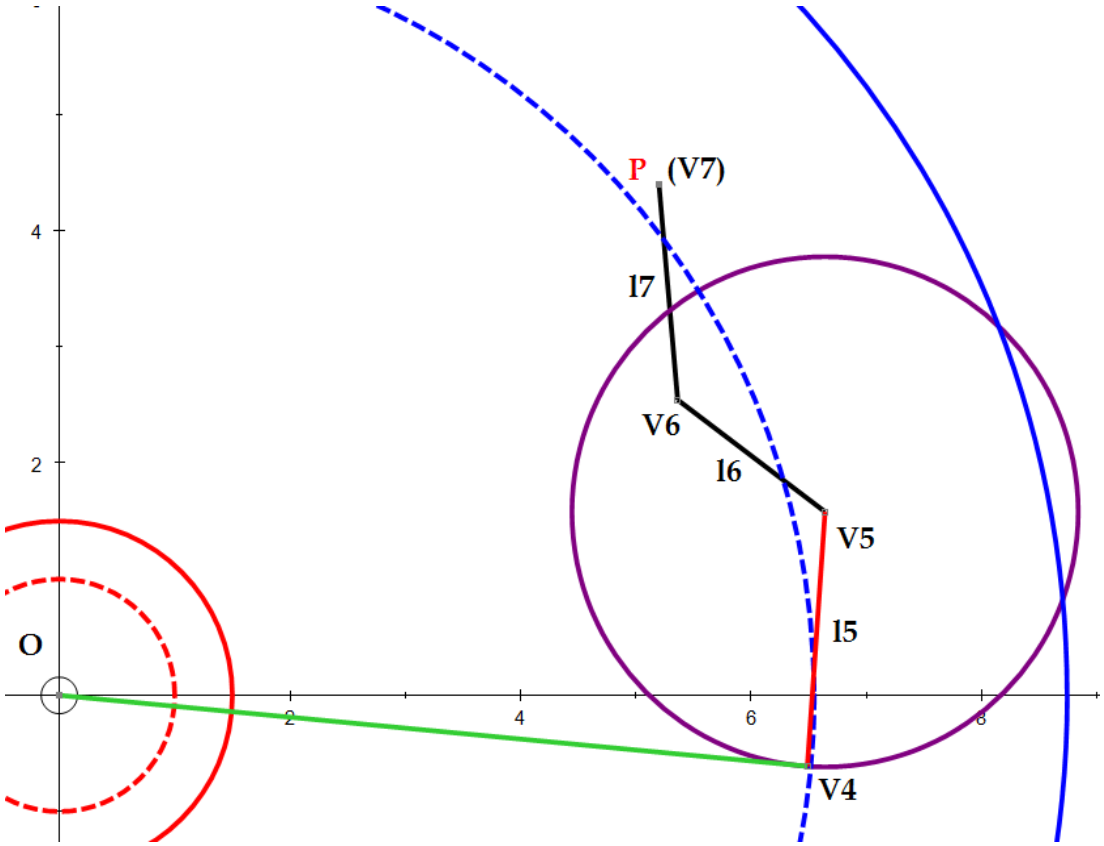

Fig 9. Case 1. Graph: the "hinge circle" (purple) intersects with the outer border of the remaining arm's R.R. (blue dashed line). Take one intersection as V_4 and fold the remaining arms as we discussed in "Reachability Region" (Green line)

Case 2. If the circle n doesn't intersect with the border of the remaining (n − 1)-segment arm's reachability region, we'd choose a point on circle n as the position of hinge v_{n-1} and iterate the same steps for v_{n-2} ... v_{n-i} until it finally comes to the first case. Notably, the point v_{n-1}'s selection has to fulfill the requirement that it's included in the remaining arm's reachability region. In this way, we can ensure the v_{n-1} is reachable and eventually get to Case 1. (A more detailed proof is given later). The complete algorithm shows as it follows. A diagram for visualization is provided below (Figure 10.).

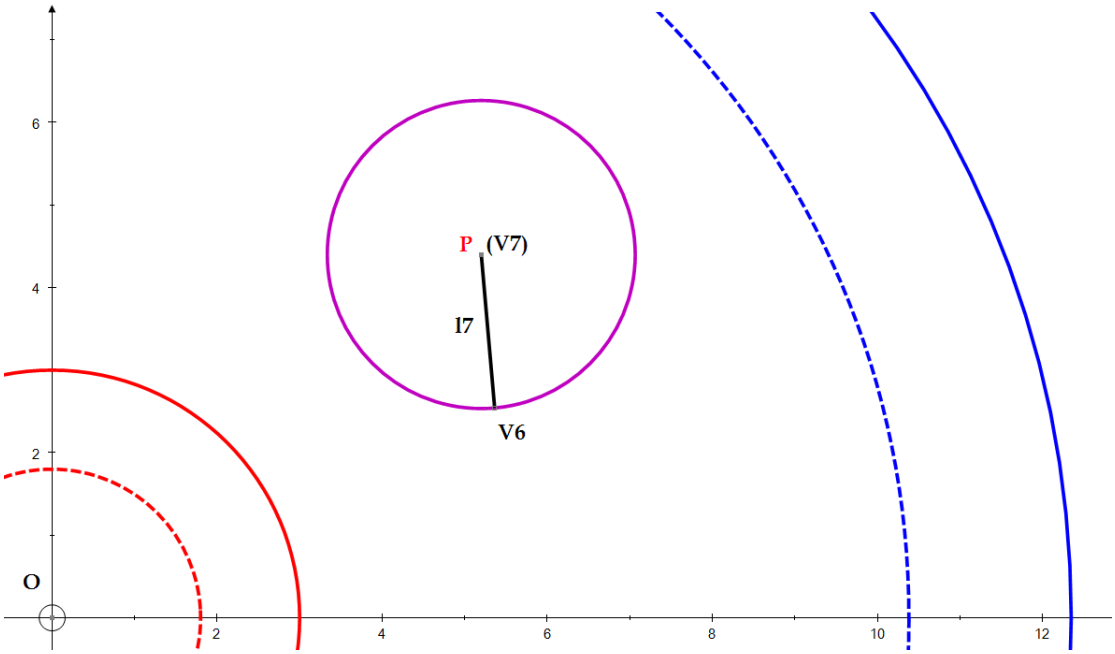

Fig 10. Case 2. Graph: the "hinge circle" (purple) doesn't intersect with the borders of arm's R.R. (red dashed line and blue dashed line). Since the "hinge circle" is included in the remaining arm's R.R., V_6 can be any point on the "hinge circle"

4.4.2 Arm-folding Algorithm

Step 1. Input the robot arm $A_n = (l_1, l_2 \dots l_n)$, the hinge set $V_n = (v_1, v_2 \dots v_n)$ and the target point $\mathrm{P}(\lambda\cos\beta, \lambda\sin\beta)$ in $\mathbb{R}^2$, where λ is the magnitude of $\vec{P}$ and β is the angle of it. Initialize the configuration set $C_n = (0,0 \dots 0)$.

Step 2. Compute the robot arm's reachability region (R. R.)

Step 3. If P is on the outer border of the arm's R.R., $C_n = (\beta, \beta, \dots \beta)$ → Stop.

Step 4. If P is on the inner border of the arm's R.R., compute the configuration so that the longest segment is opposite the remaining ones or they form a closed polygon → Stop.

Step 5. Otherwise, take P as the arm's end point v_n. Remove v_n from V_n.

Step 6. Take the last hinge v_{n-i} in V_n as the center and the last arm l_{n-i} in A_n as the radius. Construct a "hinge circle" v_{n-i}. Remove l_{n-i} from A_n and v_{n-i} from V_n (i < n).

Step 7. Compute the remaining robot arm's (A_{n-i}) reachability region ($R.R.^*$).

Step 8. If the "hinge circle" v_{n-i} intersects with the border of $R.R.^*$:

1) Outer border of $R.R.^*$: take one intersection as v_{n-i-1} and process the remaining arm A_{n-i-1} with **Step 3.** Output C_n → Stop.
2) Inner border of $R.R.^*$: take one intersection as v_{n-i-1} and process the remaining arm A_{n-i-1} with **Step 4.** Output C_n → Stop.

Step 9. Otherwise, take a point A on the "hinge circle" v_{n-i}, where A is included in $R.R.^*$; v_{n-i-1} is on A and write the new configuration into C_n.

Step 10. Repeat from **Step 6.** to **Step 9.**

4.4.3. Proof for Approach B

Problem Restatement: The Arm-folding algorithm is valid for any robot arm $A_n = (l_1, l_2 \dots l_n)$ $(n > 0)$ in $\mathbb{R}^2$ to reach a point P in its reachability region.

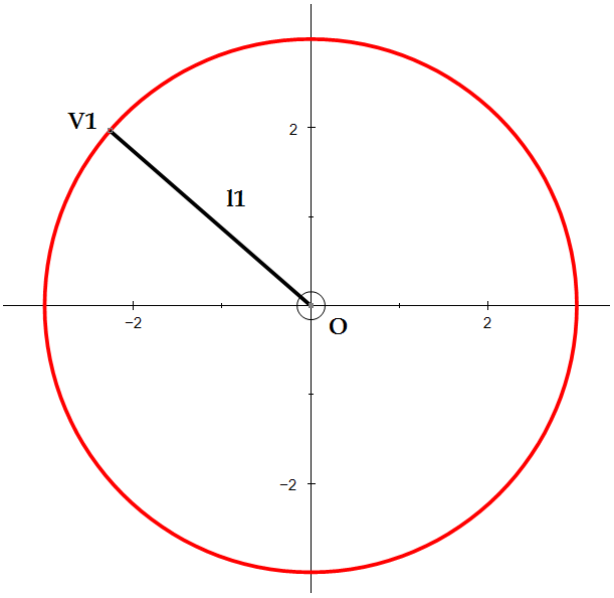

Fig 11. Base case: A_1's reachability region is only on the red border

Proof by Induction:

Base Case: When n = 1, A_n's reachability region is only the border of circle O with a radius of l_1. Thus, A_n is equivalent to the position vector of P.

In our algorithm, no matter where P is, it will recognize the point on the "outer circle" of A_n's reachability region and process to **Step 3.** The configuration is $C_1(\beta)$ and it corresponds to position vector of P's angle.

Q. E. D.

Inductive Hypothesis: Assume the Arm-folding algorithm is valid for any robot arm $A_{i-1} = (l_1, l_2 \dots l_{i-1})$ in $\mathbb{R}^2$ to reach a target point P in its reachability region.

Prove: the Arm-folding algorithm is valid for the robot arm $A_i = (l_1, l_2 \dots l_i)$

in $\mathbb{R}^2$to reach a target point P in its reachability region.

Inductive Proof: Applying the Arm-folding algorithm on the robot arm $A_i = (l_1, l_2 \dots l_i)$, since we start constructing the robot arm backwards, we want to prove: **(1)** *the algorithm will always stop for* $A_i = (l_1, l_2 \dots l_i)$

(2). v_n *is on the target point* P

(**3).** *the robot arm is continuous (i.e. connected or no gap)*

(4). *the robot arm starts from a defined origin*

1. the algorithm will always stop for $\boldsymbol{A_i = (l_1, l_2 \dots l_i)}$

Processing the Arm-folding algorithm on the robot arm $A_i = (l_1, l_2 \dots l_i)$, there're three conditions when it would stop.

First, if the target point P is on the border of A_i's reachability region, the algorithm will always stop at **Step 3.** or **Step 4.**

Otherwise, by **Step 8.** and **Step 9.**, we need to prove for a robot arm $A_n = (l_1, l_2 \dots l_n)$ $(n \geq 1)$, the new "hinge circle" v_{n-i} will always intersect with $R.R.^*$ and finally, the border of $R.R.^*$ (i.e. switch to **Step 8.**). We make the former part as **Lemma 3.**

Lemma 3. Processing the Arm-folding algorithm on robot arm $A_n = (l_1, l_2 \dots l_n)$ $(n \geq 1)$, the new "hinge circle" v_{n-i} will always intersect with $R.R.^*$ (i.e. **Step 9.** is valid)

Proof by Contradiction:

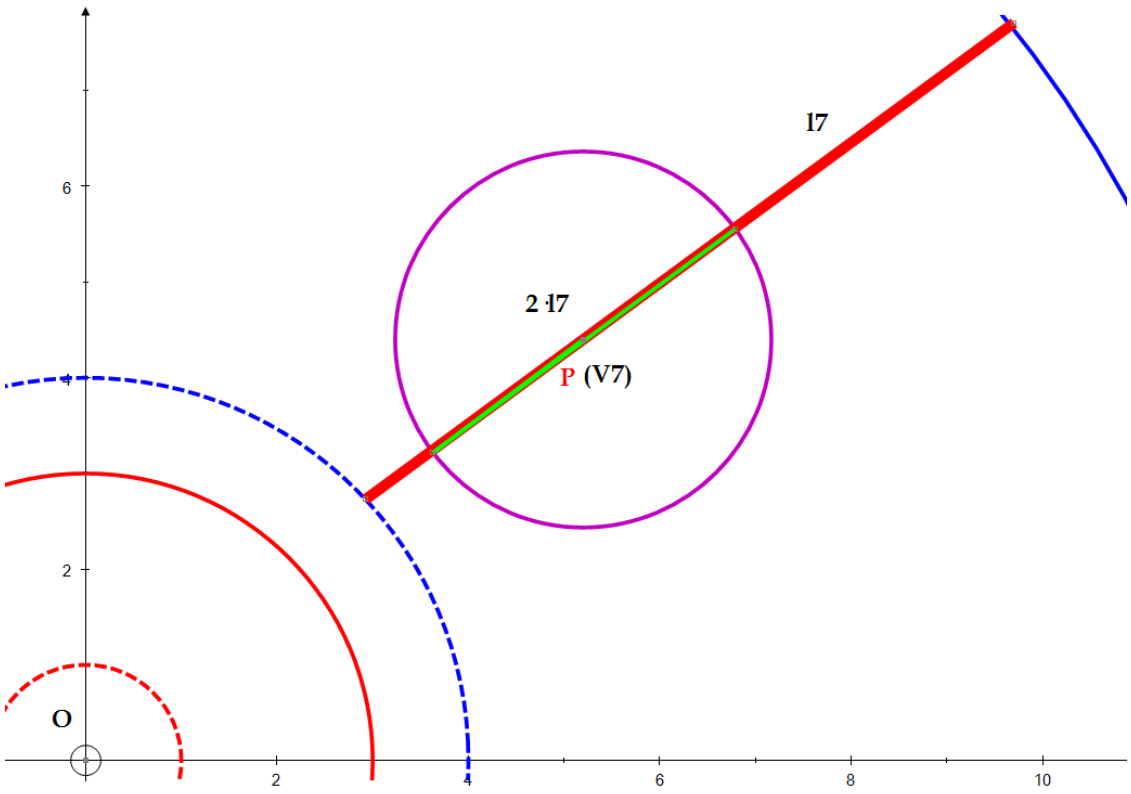

Fig 10. Proof by contradiction: the diameter of the "hinge circle" (green line) is $2 \cdot l_7$ and the decrease of the $R.R.^*$'s outer border is l_7 (red line)

Processing the Arm-folding algorithm on a robot arm $A_n = (l_1, l_2 \dots l_n)$ $(n \geq 1)$, assume v_{n-i} is included in its corresponding $R.R.^*$ but the new "hinge circle" v_{n-i} won't intersect with $R.R.^*$ That is, they're disjoint.

Denote the decrease of $R.R.^*$'s outer radius as ΔR and the diameter of the "hinge circle" v_{n-i} as d, where $d = 2 \cdot l_{n-i}$. From what we proved about the outer radius of a robot arm's reachability region, $\Delta R = l_{n-i}$.

According to our assumption, if the new "hinge circle" v_{n-i} and $R.R.^*$are disjoint, $\Delta R = l_{n-i} > d = 2 \cdot l_{n-i}$ → this is a great contradiction.

Hence, the lemma is valid any robot arm $A_n = (l_1, l_2 \dots l_n)$ $(n \geq 1)$.

Q. E. D.

By **Lemma 3.**, since the new "hinge circle" v_{n-i} will always intersect with $R.R.^*$, it may intersect with the border $R.R.^*$ in the iteration. Otherwise, as the last $R.R.^*$ (when $v_{n-i-1} = v_1$) is only the border of circle 0, the algorithm will switch to Step 8. anyway and finally stop. Accordingly, it will also stop for $A_i = (l_1, l_2 \dots l_i)$

Q. E. D.

2. v_i is on the target point P

Processing the Arm-folding algorithm on the robot arm $A_i = (l_1, l_2 \dots l_i)$, again, we'll examine all three conditions when it comes to an end.

First, if the target point P is on the border of A_i's reachability region, the algorithm will stop at **Step 3.** or **Step 4.** In these cases, as we proved before, the straightened, opposite- or polygon- folded robot arm are the only configurations for v_i to reach P. Accordingly, v_i is on the target point P. Otherwise, since the algorithm defines v_i on the target point P in Step 5, the statement is valid.

Q. E. D.

3. the output robot arm is continuous (i.e. connected or no gap)

Consider all three conditions when the algorithm comes to an end:

First, if the algorithm stops at **Step 3.** and **Step 4.**, as we proved before, the robot arm is straightened, opposite- or polygon- folded so that it reaches a point on the borders of its reachability region. In these cases, we can be sure about the output robot arm is valid. Thus, the it is connected.

Otherwise, processing the algorithm with one iteration, the problem will be reduced to the robot arm folding problem for $A_{i-1} = (l_1, l_2 \dots l_{i-1})$. Based on our inductive hypothesis, as A_{i-1} is connected and v_i is connected by **Step 9.**, A_i is connected.

Q. E. D.

4. the output robot arm starts from a defined origin

Consider all three conditions when the algorithm comes to an end:

First, if the algorithm stops at **Step 3.** and **Step 4.**, similarly, since we can be sure the output robot arm is valid as proving before, it's intuitive to conclude the output robot arm starts from a defined origin.

Otherwise, processing the algorithm with one iteration, the problem will be reduced to the robot arm folding problem for $A_{i-1} = (l_1, l_2 \dots l_{i-1})$. Based on our inductive hypothesis, as A_{i-1} starts from our defined origin and v_i is connected by **Step 9.**, A_i starts from the defined origin

Q. E. D.

Hence, as the algorithm satisfies all four requirements, it's valid for any robot arm $A_i = (l_1, l_2 \dots l_i)$ in $\mathbb{R}^2$ to reach a target point P in its reachability region.

Q. E. D.

Overall, by the principle of mathematical induction, since the Arm-folding algorithm is valid for A_1, and if it's valid for A_{i-1}, it's valid for A_i; the statement is proved for any robot arm A_n.

Q. E. D.

4.4.4 Algorithm Evaluation

For evaluation, again, we'd first consider its feasibility for computing and the efficiency with the "Big O" notation.

First, compared to the algebraic approach, this geometric algorithm is simple and intuitive for human to follow, especially when the amount of arm segments n is enormously increased. However, although practical, it'd be harder to transform the algorithm into programming languages so that computers can carry out the calculation since all the geometries have to be digitalized for machines.

Considering the computational complexity of the algorithm, we take the worst case of the algorithm and take it as the lower bound of our approach. Also, we would adopt "Big O" notation $-O(\text{"polynomial of n"})-$ for expression. It indicates that the performance of the algorithm won't be below the polynomial in terms of the size of the arm segment/hinge set, n.

Consider there're n arm segments for A_n in $\mathbb{R}^2$.

For **Step 1.** and **Step 7.**, we need to compute the reachability region of a robot arm A_i. To that end, we need to find the longest segment in each arm. Thus, we'd simply apply a most efficient algorithm known so far to sort all the segments in the descending order (divide-and-conquer algorithm)[9] and compute each of their reachability region accordingly. Here, while the sorting algorithm takes $O(n^2)$ amount of time, the remaining computation takes

[9] Cormen, Thomas H., Leiserson, Charles E. Rivest, Ronald L. "Introduction to Algorithms." MIT Press, 2000. (p.p.65)

constant time (i.e. overall, $O(n^2) + O(1) = O(n^2)$)

For the rest of steps to connect all the hinges, no matter it stops at **Step 3.**, **Step 4.**, or **Step 8.**, it will go through all the segment once and connect to the arm. Since the computation for each connection is constant time, the total amount of time for these steps is $n \cdot O(1) = O(n)$.

Overall, by summing up all the costs spent by each step of the algorithm -- $O(n^2) + O(n) = O(n^2)$ -- the lower bound of Arm-folding algorithm is $O(n^2)$. Hence, the efficiency of our geometric approach outperforms the algebraic baseline enormously. Although the language translation would be harder, it pays off especially when the amount of segment largely increases.

5. Conclusion & Reflection

To conclude, in this paper, we took a deep insight into a fundamental 1D-folding problem in $\mathbb{R}^2$, namely the Robot Arm's folding problem. While we investigated the capacity (reachability region) for general robot arm in $\mathbb{R}^2$, two approaches, both algebraically and geometrically, are proposed as solutions.

Weighing the pros and cons of each approach, we consider the algebraic approach as an intuitive baseline for the problem with low computational efficiency. However, it'd be highly preferred if the amount of arm segments are limited, and it can be easily translated to programming languages. For our improved method, it's robust for both human and machine to compute with a decent computational complexity of $0(n^2)$. Still, certain concepts are not well-defined in the detailed description for machines to understand, so further more work has to be done for clarification. Notably, in this process of implementation, a considerable computational complexity may be added.

Considering some future work that can be made, we're particularly interested in two branches.

First, to generalize our problem, we'd like to tackle 2-D folding problem in $\mathbb{R}^3$. In this case, the robot arms will become plates joined by hinges, similarly to what we play in Origami. Thus, one solution may be mapping the Origami math to the problem and seeing what the reaction will be. Another potential approach is to take "a slice" of $\mathbb{R}^3$ containing the origin and the target point

P; then we may reduce it to the Robot Arm's folding problem!

Another direction is to see the minimum "non-straight bends" we really need for an arbitrary arm A_n to reach P. The ultimate goal of this question may prove mathematically why human arm has three segments after the Natural Selection.

Again, it should be emphasized that the work of Robot Arm's folding problem with high efficiency will boost the development of various fields such as manufacturing industry, space exploit and so on. A smart mathematics algorithm will even alleviate many engineers' concern when designing. Although the computational complexity of the algorithm proposed in this paper may be incomparable to other main-stream approaches, we hope to hint more mathematicians to solve the problem in the future.

6. Bibliography

[1] O'Rourke, Joseph. Demaine, Erik D. *"Geometric Folding Algorithms: Linkages, Origami, Polyhedra."* Cambridge University Press, First published 2007.

[2] Fannon, Paul. Kadelburg, Vesna. Woolley, Ben. Ward, Stephen. "*Mathematics Higher Level for the IB Diploma.*" Cambridge University Press, First published 2012.

[3] Devadoss, Satyan D. O'Rourke, Joseph. "Discrete and Computational Geometry." Princeton University Press, 2011.

[4] O'Rourke, Joseph. "*How to fold it.*" Cambridge University Press, 2007.

[5] Cormen, Thomas H., Leiserson, Charles E. Rivest, Ronald L. "Introduction to Algorithms." MIT Press, 2000.

[6] Weisstein, Eric W. "Parallelogram Law." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/ParallelogramLaw.html

[7] Weisstein, Eric W. "Triangle Inequality." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/TriangleInequality.html

[8] Weisstein, Eric W. "Polar Coordinates." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/PolarCoordinates.html