Growing islands to approximate tessellations of the plane by Voronoi diagrams

Christopher Schiffmann
Department of Advanced Computing Sciences
Faculty of Science and Engineering
Maastricht University
Maastricht, The Netherlands

***Abstract*—Voronoi diagrams are well-studied objects in the field of computational geometry. One problem related to Voronoi diagrams is the Inverse Voronoi problem where, given a tessellation of the plane into convex regions, we wish to find the points which could have generated the tessellation. Sometimes this is not possible, but it can be desirable to approximate the original tessellation by a Voronoi diagram. This paper presents a new algorithm for generating these approximations using a proposed definition for "Voronoi-ness" of arbitrary tessellations, previously not well-defined in the literature. The results show that the algorithm does well on tessellations with mostly well-shaped and uniformly sized regions. The algorithm also marginally improves an existing solution for the approximation of territories of *Tilapia mossambica* by a Voronoi tessellation.**

I. Introduction

Voronoi diagrams, also known as Voronoi tessellations and Dirichlet tessellations, are well-studied objects in the field of computational geometry. In the 2-dimensional unweighted case it consists of non-overlapping convex regions generated by a set of generator points. The Voronoi tessellation has the property that every point inside a region is closer to that region's generator point than that of any other region. In the context of urban planning, we could consider schools or other public facilities to be the generator points of a Voronoi diagram overlapping a city to determine the closest neighborhoods to these facilities.

Associated with the study of Voronoi tessellations is the *inverse Voronoi problem*: given a set of regions, we are asked to find a set of generator points which could have generated them. This is also known as reconstructing the Voronoi tessellation, which has been studied as well, for example in by Ash & Bolker [2]. Heath & Kasif [7] showed that this is an easier version of the minimal Voronoi cover problem which is NP-Hard, and found an efficient algorithm to solve the inverse Voronoi problem.

However, in some cases it may be possible that no such set of generator points exists. This may be the case if, for example, we wish study a tessellation from nature which may have stemmed from a uniform growth process perturbed by exterior forces. Most examples stemming from nature fall under this category, e.g., cell structures. Because of this, it is also of interest to study approximations of arbitrary tessellations by Voronoi tessellations. Algorithms for this have been proposed by Suzuki & Iri [9] and Aguilera et al. [1].

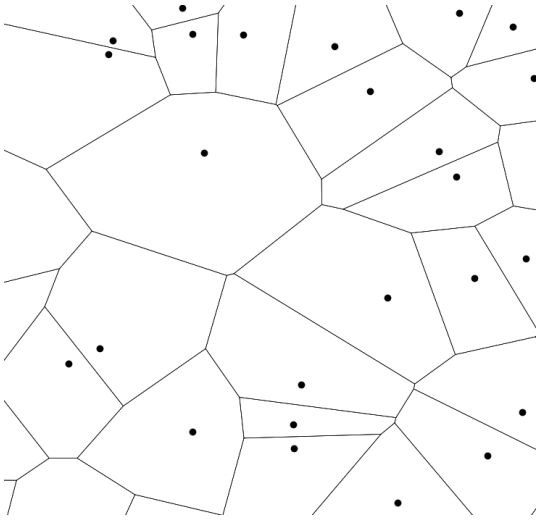


Fig. 1. Example of a Voronoi tessellation, where the generator points are marked by dots, and their corresponding regions have thin outlines.

Something that so far has not been well-defined in the literature however is an idea of "Voronoi-ness", i.e., determining how close a given arbitrary tessellation is to bearing properties of Voronoi tessellations. A motivation for such a definition would be knowing when it may not make sense to make such an approximation by a Voronoi tessellation, for example if it is known in advance that the input tessellation is "far" from being a Voronoi tessellation. Honda [8] expresses results in terms of a angle-based deviance value, and Suzuki & Iri [9] use an area discrepancy measure, but both of these require the existence of an input & an output tessellation to calculate these values. Ideally, we would like a measure which we can apply to a single tessellation without requiring another one to compare to.

This paper proposes a measure for "Voronoi-ness" and an algorithm making use of this idea to approximate planar tessellations by Voronoi tessellations to determine the applicability

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. Supervisor: Steven Chaplick

of the theoretical framework behind this measure. To this end, we ask the following research questions:

How can Voronoi-likeness of a tessellation be measured? In other words, we want to determine some kind of measure which tells us how well a Voronoi tessellation can be used to approximate an arbitrary planar tessellations.

Can this measure be efficiently computed? Assuming the measure can be computed exactly, it would be ideal to have an efficient algorithm for this.

Can this measure be efficiently approximated? In the event where the proposed measure can't be exactly computed, it may be desirable to instead approximate it, with an emphasis on efficient computation as in the previous research question.

How can the measure be used to create approximations of arbitrary tessellations by Voronoi tessellations? Once we determine the measure, we want to apply it to the use case of generating approximations of arbitrary tessellations by Voronoi diagrams, the motivation of which has been outlined before.

This paper is structured as follows: First, a mathematical framework for addressing the problem is developed in section II, then an algorithm using this framework is presented in section III. This algorithm is then put to the test in two different experimental setups - first, applying the algorithm to the Inverse Voronoi Problem, and secondly, using the algorithm to approximate tessellations not guaranteed to be Voronoi tessellations.

A Python implementation of the algorithm presented in this paper can be found on the following GitHub repository: `https://github.com/cjphs/Voronoi-Likeness`

II. Preliminaries

In this section, a mathematical foundation is described for addressing the problem at hand, beginning with some definitions, a general framework, and a formal problem formulation, all leading up to the proposed Voronoi-ness measure.

A. Definitions

In order to develop the framework for this problem, the following definitions are needed:

Definition II.1 (Region)**.** *A region is a set of points forming a convex polygon with a finite amount of sides.*

Definition II.2 (Tessellation)**.** *A tessellation T is a tuple (R, V) of regions in R with no overlaps and gaps, and vertices in V. Each point in V is the intersection of three or more regions in R.*

We furthermore let $n = |R|$, and $m = |V|$.

Definition II.3 (Voronoi tessellation)**.** *A Voronoi tessellation is a tessellation $T = (R, V)$ for which there exists a set of points $P = \{p_1, p_2, \ldots, p_n\}$ $(p_i \in R_i \ \forall R_i \in R)$ such that:*

$$R_k = \{x \in X \mid d(x, p_x) \leq d(x, p_j) \text{ for all } j \neq k\} \quad (1)$$

From equation 1, we can define $\mathcal{V}(P) : P \to T$ to be the function which generates the Voronoi tessellation of the set of points P.

B. Framework

To properly formulate the problem addressed in this research, a general framework needs to be defined first.

1) Voronoi tessellation criteria: Let $T = (R, V)$ be a Voronoi tessellation with n regions and m corner vertices. In order to determine if a set of points $P = \{p_1, p_2, ..., p_n\}$ is a valid set of generator points for T, i.e., if $\mathcal{V}(P) \equiv T$, we want to find some minimal conditions which allow us to check this with few computations. If a region R_i was generated by a point p_i, then the corner vertices of R_i are the furthest points in the region from p_i, and any point on the region's edges is closer to p_i than the furthest corner vertex of a particular edge. Illustrated in Figure 2 and with a quick proof:

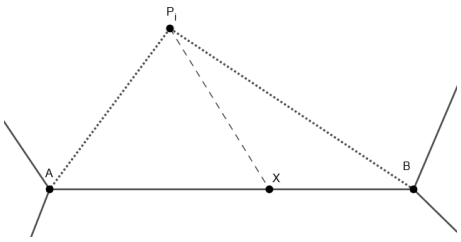


Fig. 2. An edge segment AB of a Voronoi region R_i generated by P_i. $PA \leq PB$, and $PX \leq PB$.

Proposition II.1. *Let PAB be a triangle such that $PA, PB, AB > 0$. Let X be on (AB) such that $AB = AX + XB$, where $AX, XB \geq 0$. Then $PX \leq PB$.*

Proof.

$$\begin{array}{lll} PX & \leq PB & \mid PB < PA + AB \\ \iff PX & < PA + AB & \mid PX > PA - AX \\ \iff \cancel{PA} - AX & < \cancel{PA} + AB & \mid \\ \iff -AX & < AB & \mid + AX \\ \iff 0 & < AB + AX & \mid \end{array}$$

Since $AB > 0$ and $AX \geq 0$, $AB + AX \geq 0$, and thus $PX \leq PB$. □

Following the definition of a Voronoi tessellation as shown in Equation 1, any point closer to the region's generator point than the edge segments is included in the region. It suffices to check that for each corner vertex of the tessellation, the generator points of its adjacent regions are closer (equally close w.r.t each other) to the corner vertex than any other generator point of the tessellation.

2) Spokes and islands: In [1], Voronoi tessellations are used to approximate imprecise planar tessellations, which differ from Definition II.2 with the existence of gaps between all regions, i.e., no two regions share any points. These disjointed regions can be thought of as islands separated by waters in form of the gaps.

Similarly, we can consider islands to be scaled versions of the tessellation's regions as a relaxation of the Voronoi tessellation criteria described previously. The motivation here

is to scale the regions of the tessellation by some factor $\omega \in [0, 1]$ along spokes drawn from the corner vertices of the regions to their generator points. The largest possible scaling of every region in the tessellation could serve as a measure for Voronoi-likeness expressed by ω, and the larger that the islands can be scaled, the closer the tessellation resembles a Voronoi tessellation.

Figure 3 shows an example of a region scaled by $\omega = 0.5$ with the corresponding spokes to the generator point and the island given by the scaled region.

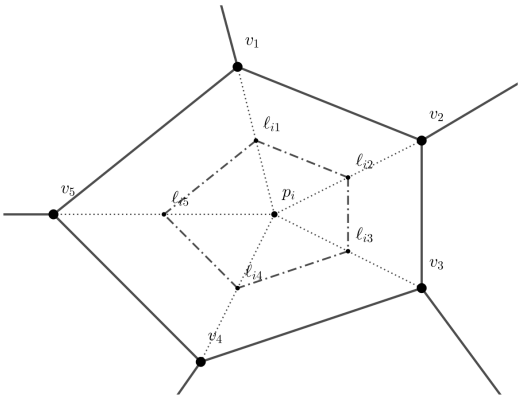


Fig. 3. Example of a region R_i scaled by $\omega = 0.5$ along the spokes connecting p_i to the region's corner vertices.

Since this paper involves tessellations not guaranteed to be Voronoi tessellations, the central point of the spokes is still referred to as the generator point even if the tessellation in question is not a Voronoi tessellation. In this context, the generator point generates the spokes & the corresponding region's island, hence the naming still applies.

The corner vertices of the islands are defined as follows:

$$\ell_{ij} = p_i + \omega(\overrightarrow{p_i v_j}) \quad \forall \ell_{ij} \in R_i. \tag{2}$$

Note that if $\omega = 0$ then $\ell_{ij} = p_i \ \ \forall\ v_j \in R_i$. If $\omega = 1$ then $\ell_{ij} = v_j \ \ \forall\ v_j \in R_i$.

Through the rest of this paper, these scaled corner vertices are referred to as label points.

3) Classifying the label points: As laid out in Section II-B1, we are interested in the distances between corner vertices and generator points. Because of this, it is of interest to have some kind of classification for the label points based on which generator point is closest. A label here is simply the index of a region since we already denote regions as $R = \{R_1, R_2, \ldots, R_n\}$ and generator points as $P = \{p_1, p_2, \ldots, p_n\}$ where $p_i \in R_i$, hence the naming of the label points.

We define a function $L : \ell \to \mathbb{N}$ as follows:

$$L(\ell_{ij}) = \begin{cases} i & \text{if } d(p_i, \ell_{ij}) \leq d(p_{k\neq i}, \ell_{ij}) \\ \arg\min_k d(p_k, \ell_{ij}) & \text{otherwise} \end{cases} \tag{3}$$

where $d(p, q)$ is the Euclidean distance between two points p and q. $L(\ell_{ij})$ finds the region index of the nearest point in P to ℓ_{ij}, so if $L(\ell_{ij}) = i$, then p_i is closer to $L(\ell_{ij})$ than any other point in P.

If $L(\ell_{ij}) = i \ \ \forall \ell_{ij}$, then we say that the tessellation has a satisfied labeling for ω. In other words: If we generate $\mathcal{V}(P)$ and if the label points of the island belonging to region R_i are inside the region generated by p_i in $\mathcal{V}(P)$, then T has a satisfied labeling for ω and the set of candidate generator points P.

Additionally, if we have a set of generator points P for which $T \equiv \mathcal{V}(P)$, then T has a satisfied labeling for all values of $\omega \in [0, 1]$.

C. Problem formulation

Having defined a general framework, we can give a formal definition of the problem.

If we have a tessellation and a set of generator point candidates P, we want to know how we can scale the regions of the tessellation by ω such that there is a satisfied labeling for all label points in the tessellation. In the case where we find $\omega = 1$, then $T \equiv \mathcal{V}(P)$.

Problem 1: Given $T = (R, V)$ and $P = \{p_1, p_2, ..., p_n\}$, find the largest $\omega \in [0, 1]$ such that $L(\ell_{ij}) = i \ \ \forall \ell_{ij}$ (assuming such an ω exists).

From this follows the optimization problem where we do not have a set of candidate generator points and we want to find the largest value of ω for which a set of points yielding a satisfied labeling exists.

Problem 2: Given $T = (R, V)$, find the largest $\omega \in [0, 1]$ for which a set of points $P = \{p_1, p_2, ..., p_n\}$ can be found such that $L(\ell_{ij}) = i \ \forall \ell_{ij}$, i.e., $d(p_i, \ell_{ij}) \leq d(p_{k\neq i}, \ell_{ij}) \ \forall p_i \in P$.

If we find a solution for this problem, then ω can be used a measure for how close the tessellation T is to being a Voronoi tessellation. Then, ω defines how the regions of T can be scaled to preserve Voronoi-like distance properties at the corners of the islands since, in the case that the given tessellation indeed corresponds to a proper Voronoi tessellation, the corner vertex where three regions meet should be equidistant to each adjacent region's generator point.

Throughout the rest of this paper, the maximal value of ω fulfilling the criteria of Problem 2 will be referred to as the ω-measure of a tessellation and serves as this paper's proposed measure for Voronoi-likeness.

III. Methodology

Having defined the framework and a proper formulation for the problem at hand, we use these to develop an algorithm to approximate arbitrary tessellations by Voronoi tessellations in the two-dimensional unweighted case. First, a theoretical

description of this algorithm is given, and then an implementation of it is described. Lastly, a measure to compute the accuracy of the resulting approximations is described.

A. Approximating tessellations

For the purpose of approximating tessellations by Voronoi tessellations, a local search approach will be used, similarly to [9] and [1]. As input the algorithm should receive a tessellation, and as output it should give a set of candidate generator points for the input tessellation. If the input does not correspond to a Voronoi tessellation, then the output generator points should be able to generate a Voronoi tessellation which somewhat resembles the input. Applying a solution of Problem 1 to the output generator points yields a lower bound for the ω-measure of the input tessellation.

The principle idea is as follows: Given a tessellation $T = (R, V)$, start with an initial guess for the tessellation's generator points $P^{(0)} = \{c_1, c_2, \ldots, c_n\}$, where c_i is the centroid of region R_i. Using the solution to Problem 1, compute the maximal value of ω for which the tessellation has a satisfied labeling with the initial guess. Then, set a new target value for ω, and move the generator point guesses according to the labels of the label points generated using this target value. This process is then repeated with the moved candidate generator points.

We want $L(\ell_{ij}) = i$ for all label points. Taking the condition $d(p_i, \ell_{ij}) \leq d(p_{k\neq i}, \ell_{ij})$ from equation 3, the following solution for Problem 1 can be derived:

$$\omega \leq \begin{cases} \frac{\Theta}{\Psi} & \text{if } \Psi > 0 \\ 1 & \text{if } \Psi = 0 \\ -\frac{\Theta}{\Psi} & \text{if } \Psi < 0 \end{cases}$$

where:

$$\begin{aligned} \Theta =& p_{kx}^2 + p_{ky}^2 - p_{ix}^2 - p_{iy}^2 - \\ & 2p_{ix}(p_{kx} - p_{ix}) - 2p_{iy}(p_{ky} - p_{iy}) \\ \Psi =& 2\left[(v_{jx} - p_{ix})(p_{kx} - p_{ix}) + (v_{jy} - p_{iy})(p_{ky} - p_{iy})\right] \end{aligned}$$

$\forall R_i, v_j, p_{k\neq i}$.

The derivation for this can be found in section VIII-A.

From this, let $\Omega : T, P \to \mathbb{R}$ be the function which determines the highest possible value for ω given the current arrangement of generator points P in the tessellation T, yielding a solution for Problem 1. Figure 4 shows an example of a tessellation with candidate generator points where the the regions are scaled by $\Omega(T, P)$ such that the corner vertices of the islands are closer to the region's generator point than that of any other region. The Voronoi tessellation generated by P is overlaid in blue. If ω were any larger, then some of the label points would fall into the wrong regions in $\mathcal{V}(P)$, i.e., $\mathcal{V}(P)$ constrains $\Omega(T, P)$.

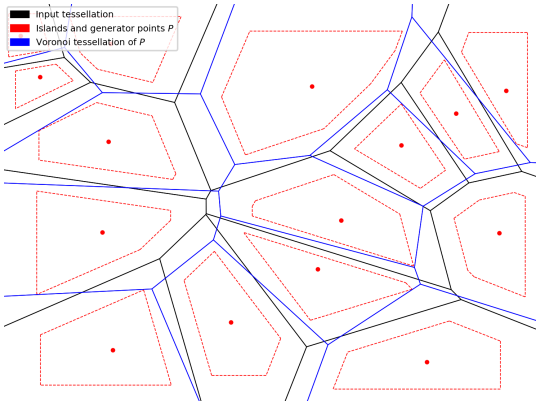


Fig. 4. Regions of a tessellation scaled by $\omega = \Omega(T, P)$, where P is the set of candidate generator points marked by red dots, overlaid by the Voronoi tessellation of P $\mathcal{V}(P)$

For the step where the generator points are moved, we recall again that we want $L(\ell_{ij}) = i$ for all label points ℓ_{ij}. The label points of each region should be closer to that region's generator point than that of any other region, so we move the generator points based on the labeling of the label points. If a label point ℓ_{ij} is in region R_i but is closer to $p_{k\neq i}$ than p_i, then ℓ_{ij} should push p_k away and pull p_i towards itself. Equation 4 provides a mathematical description of this for an iterative approach for each iteration $\aleph \in \mathbb{N}$:

$$p_i^{(\aleph+1)} = p_i^{(\aleph)} + \phi_i \left(\sum^{x} \overrightarrow{p_i^{(\aleph)} x} + \sum^{y} \overrightarrow{y p_i^{(\aleph)}} \right) \tag{4}$$

$$\forall x, y \text{ such that } x \in R_i \text{ and } L(x) \neq i, \quad y \notin R_i \text{ and } L(y) = i.$$

The parameter ϕ_i controls the amount of nudging happening within region R_i. This can be fine-tuned. For instance, it may be more ideal to dampen the nudging in smaller regions. Through the rest of this paper however, we will be using only one value for all ϕ_i. Since the goal is to obtain $L(\ell_{ij}) = i$ for all label points in the tessellation, we perform this operation until no more points are nudged, i.e. we reach an iteration k such that $p_i^{(k)} = p_i^{(k-1)} \quad \forall i$. Additionally, if $\Omega(T, P^{(k)}) = 1$, then the set of generator points $P^{(k)}$ is also a set of generator points for the original tessellation T, and $T \equiv \mathcal{V}(P)$.

B. Algorithmic implementation

In this section, an algorithm for approximating arbitrary tessellations by Voronoi tessellations is presented, making use of the previously-described framework.

Implementation of equation 4:

Algorithm 1 Nudging generator point guesses by ϕ

Input Label points $\mathcal{L}$, current generator points P, ϕ
Output Nudged generator points P'.

1: $\nabla_i \leftarrow \overrightarrow{0}\ \forall i$
2: **for** $\ell \in \mathcal{L}$ **do**
3: p $\leftarrow$ nearest point in P to ℓ
4: i $\leftarrow L(\ell_{ij})$
5: k $\leftarrow$ region index of p
6: **if** $k \neq i$ **then**
7: $\nabla_i \leftarrow \nabla_i + \overrightarrow{p_i \ell}$
8: $\nabla_k \leftarrow \nabla_k + \overrightarrow{\ell p_k}$
9: **end if**
10: **end for**
11: **for** $p_i \in P$ **do**
12: $p_i' \leftarrow p_i + \phi \nabla_i$
13: **end for**
14: **return** P'

Algorithm 1 is then used as a subroutine in the main approximation algorithm:

Algorithm 2 Voronoi tessellation approximation

Input Tessellation $T = (R, V)$
Output Approximate generator points

1: $P \leftarrow$ centroids of regions in R
2: done $\leftarrow$ False
3: $\phi^* \leftarrow \phi$
4: **while not** done **do**
5: $\omega \leftarrow \Omega(T, P)$ **(*)**
6: $\omega' \leftarrow \omega + (1 - \omega)/2$
7: $\phi \leftarrow \phi^*(1 - \omega)$ **(**)**
8: **while not** all labels satisfied **do**
9: $L \leftarrow$ generateLabelPoints(ω')
10: $P \leftarrow$ nudgeGeneratorPoints(P, L, ϕ)
11: **if** $L(\ell_{ij}) = i\ \forall \ell_{ij}$ **then**
12: all labels satisfied $\leftarrow$ True
13: **else**
14: **if** solution not improving **then (***)**
15: $\omega' \leftarrow (\omega' + \omega)/2$
16: **end if**
17: **if** $\omega' - \omega < \epsilon$ **then (****)**
18: done $\leftarrow$ True
19: **break**
20: **end if**
21: **end if**
22: **end while**
23: **end while**
24: **return** P

In essence, Algorithm 2 tries to optimize the set of generator points P such that $\Omega(T, P)$ gradually increases. In each iteration $\aleph$, a target value $\omega'^{(\aleph)}$ is set as the mid-way point between $\Omega(T, P^{(\aleph)})$ and 1 in an exponential search since if the input tessellation corresponds to a Voronoi tessellation, then we would ideally like to find P such that $\Omega(T, P) = 1$. The generator points are moved according to Algorithm 1 until all label points generated using ω' have a correct labeling, i.e., $L(\ell_{ij}) = i\ \forall \ell_{ij}$. If the algorithm struggles with the nudging process for ω', which could happen if the input tessellation has a ω-measure < 1, or if the current guess of generator points is still too distant from the actual optimum, then ω' is reduced to the midway-point between itself and ω.

Certain lines of the pseudo-code are noted for further elaboration:

- **(*)**: This is where the solution for problem 1 comes into play. For the current guess of generator points, which to begin with is initialized to the centroids of all regions, the maximum value of ω is computed for which all label points have a correct labeling. Typically, $\Omega(T, P^{(\aleph)}) \geq \omega'^{(\aleph-1)}$, providing a minor boost to the approximation process.
- **(**)**: The closer ω gets to 1, the more we dampen ϕ, as we wish not to overdo the nudging process as the algorithm's solution improves. $\phi*$ here is simply the initial input value of ϕ, and is stored before the algorithm runs (line 3). This step aids in the convergence of the algorithm, as we wish to not overdo the nudging step when ω gets closer to 1. A similar technique is used in force-directed drawing of graphs, where the monotonic convergence of majorization ensures that the algorithm eventually reaches a local minimum and stops ([5]).
- **(***)**: There exist different options for the criteria of solution improvement. In the algorithm's implementation for this paper's experiments, the solution improvement criteria is naively defined as whether or not the amount of satisfied label points increases, i.e., if the algorithm stagnates and reaches a plateau with the current target value of ω, the target is reduced.
- **(****)**: To ensure that the algorithm terminates at some point, we stop if ω' is reduced close to ω within some margin ϵ, chosen here to be 0.0001.

Using the outputted set of generator points P, $\Omega(T, P)$ yields a lower bound for the ω-measure of the input tessellation T.

As for complexity, the algorithm's main bottleneck is the calculation of $\Omega(T, P)$ at the beginning of each parent iteration. In its current implementation, the computation of this step requires $\mathcal{O}(kn^2)$ time, where k is the number of corner vertices in the tessellation. This could be sped up by applying a more efficient data structure for storing the positions of the generator points & corner vertices, for instance using a range tree. The trees for querying would have to be rebuilt in each parent iteration due to the movement of the generator points, but this would ultimately yield a runtime in $\mathcal{O}(n \log n)$ for each corner vertex, and a query time of $\mathcal{O}(\log n + k)$, where k is the number of points within the query interval ([4]).

An example of several steps of the algorithm's process is shown in Section VIII-B.

C. Area discrepancies

In order to determine the accuracy of the approximations yielded by Algorithm 2, we want a measure that indicates how the output differs from the input. For this, the discrepancy measure introduced in [9] is used, where the discrepancy between two tessellations is defined by the area of the intersection between regions of different indices, i.e., the area where regions overlap with regions generated by different generator points than their own. Figure 5 shows an example of this discrepancy between two tessellations where one was generated using the centroids of the other. The discrepancy is marked by the shaded areas.

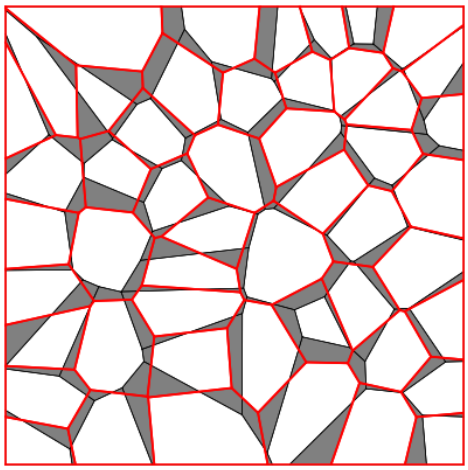


Fig. 5. An example of area discrepancy between two tessellations (represented by the shaded areas).

Mathematically, this discrepancy is defined as follows:

$$\delta(T, A) = \cup_{i,j \neq i}(R_i \cap R'_j) \tag{5}$$

where $T = (R, V)$ is the input tessellation, and $A = (R', V)$ is the Voronoi tessellation obtained by the generator points yielded by the iterative approximation method.

IV. Experiments

This section outlines the experiments that were conducted to test the algorithm's practicality. These experiments are divided into two main parts. In the first part, the algorithm attempts to find the generator points of proper Voronoi tessellations; in the second part, it attempts to generate approximations for tessellations not guaranteed to be Voronoi tessellations.

A. Approximating Voronoi tessellations

In order to find a decent value for ϕ in equation 4, the algorithm is tested on 100 randomly-generated Voronoi tessellations with different hand-picked values. Each tessellation was generated using 32 points. For each value of ϕ, we are interested in the resulting median values for ω (the lower bound of the tessellation's Voronoi-ness) and the area discrepancy δ between the input tessellation and the algorithm's output, as well as the median runtime for each tested value.

B. Approximating arbitrary tessellations

To test the algorithm's ability of approximating arbitrary tessellations, we consider two cases here. Since the algorithm sets the initial guess for the tessellation's generator points to each region's centroids, it is of interest to examine if the algorithm performs better on tessellations which could be Voronoi tessellations where the generator points correspond to the regions' centroids (e.g., a hexagonal grid).

For this experiment, we take such a Voronoi tessellation and increasingly distort the positions of its vertices while making sure not to destroy the convexity & non-overlapping properties of every region. For each of these distortion values, we run the algorithm fifty times on randomly-distorted input generations using the best value for ϕ from the first experiment. Again, we are interested in the resulting average values for ω and δ.

Lastly, the algorithm is applied to approximate figure 1 from [6], which depicts territories[1] formed by *Tilapia mossambica*, a species of fish native to southeastern Africa, present here in Figure 6. The algorithm presented in [9] yields an area discrepancy of 3.8% with their solution, with which they conclude that the tessellation may be approximately regarded as a Voronoi diagram. So it is of interest to see if the approximation algorithm presented in this paper yields a better result.

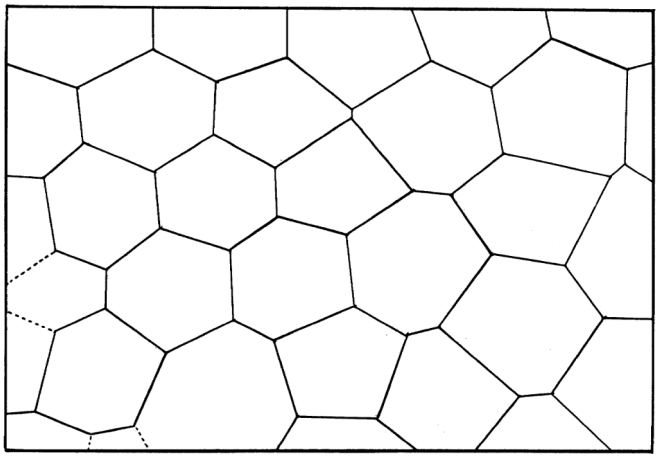


Fig. 6. Figure 1 from [6], depicting territories formed by *Tilapia mossambica*.

These experiments were conducted on a personal computer with an Intel i5-9300H processor and 16 GB RAM running on Windows 10.

[1]In [6], a territory is defined as an area defended by an animal / a group of animals. In this case, Figure 6 shows the territories formed by multiple different groups of *Tilapia mossambica*. Barlow [3] found that these territories are mostly polygonal.

V. Results

In this section, the results of the aforementioned experiments are shown. The purpose of each figure is described beforehand as a summary of the key details mentioned in section IV.

For the experiment of determining a good value for ϕ in the context of the Inverse Voronoi problem, several test values were hand-picked and then tested on 100 different randomly-generated Voronoi tessellations. Figure 7 shows one of the output tessellations along with the obtained generator points laid over the input tessellation and its original generator points.

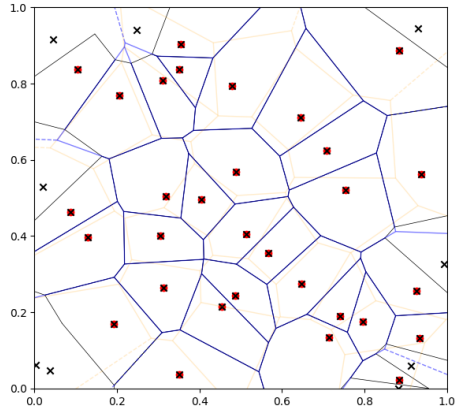


Fig. 7. Generator points of input diagram rediscovered by the algorithm with $\omega = 0.9999905$.

For each of these hand-picked values for ϕ, the results for ω and δ over 100 runs on the random Voronoi tessellations are summarized by box-and-whisker plots in Figure 8 to get an idea of the spread of the algorithm's accuracy.

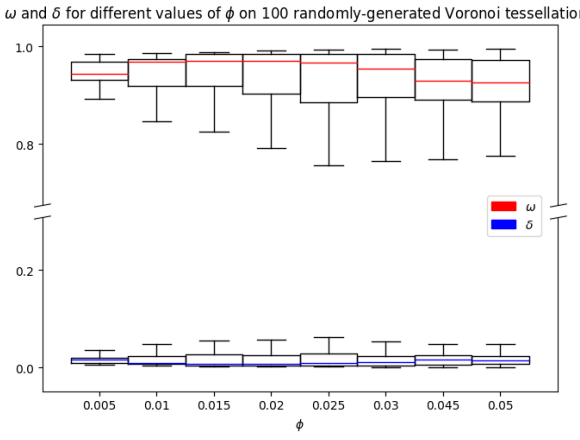


Fig. 8. Box-and-whisker plots for the overall effect of ϕ on ω and δ of the algorithm's output over a set of 100 randomly-generated Voronoi tessellations of 32 generator points each

To determine a decent compromise between runtime and quality of the output, the trajectory of the relationship between median execution time (in seconds) and the median area discrepancy is plotted for each tested value of ϕ in Figure 9.

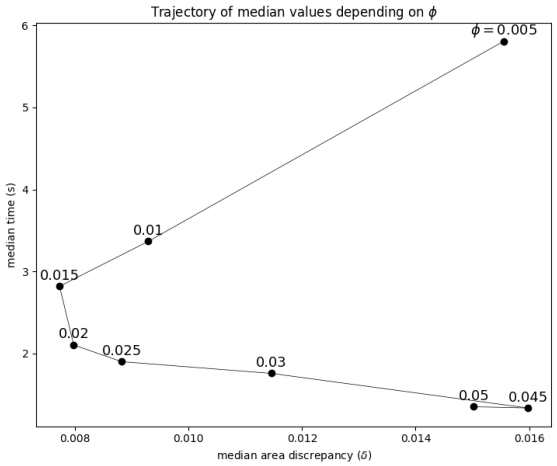


Fig. 9. Trajectory of the relationship between the median processing time in seconds and the median area discrepancy as the test value for ϕ increases.

To see how the algorithm fares on tessellations not guaranteed to be Voronoi tessellations, the vertices of a hexagonal tiling are increasingly distorted. Figure 10 shows an example of a distorted hexagonal tiling with a Voronoi tessellation approximation laid over in blue.

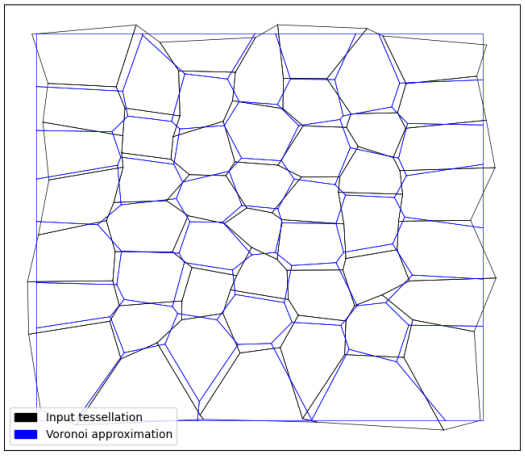


Fig. 10. Approximation of a slightly distorted hexagonal Voronoi tessellation with a vertex distortion of 3% using $\phi = 0.02$

Similarly to Figure 8, the algorithm is tested on 50 different increasingly-distorted tessellations. The results of this are summarized by a box-and-whisker plot in Figure 11.

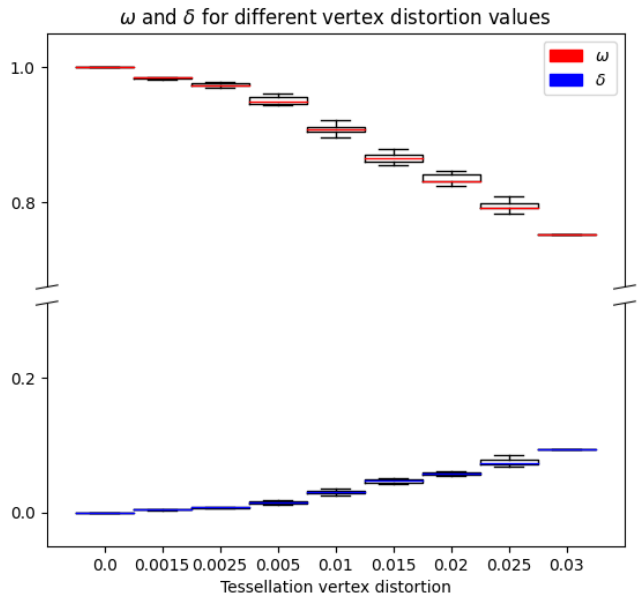


Fig. 11. Box-and-whisker plots for ω and the area discrepancy δ for each tessellation vertex distortion factor using $\phi = 0.02$

Finally, the algorithm is used to generate a Voronoi tessellation approximation of the territories of *Tilapia mossambica*, as is done in [9]. Their approximation yields an area discrepancy of 3.8%, while the algorithm presented in this paper achieves an area discrepancy of 3.08%.

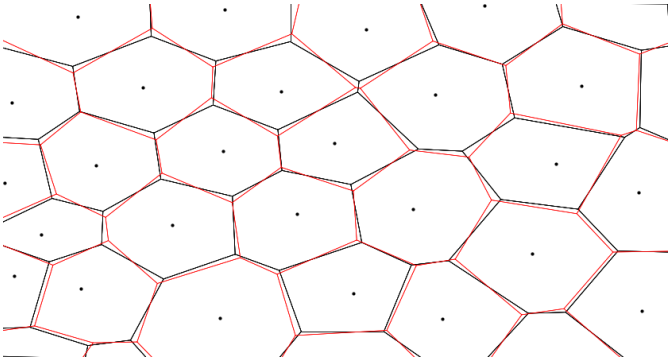


Fig. 12. An approximation of figure 1 from [6] with an area discrepancy of 3.08% using $\phi = 0.02$.

VI. Discussion

As described in the methodology, this paper proposes a value of "Voronoi-ness" for a tessellation in terms of a value $0 \leq \omega \leq 1$ which indicates how the regions of a tessellation can be scaled in order for there to exist a set of points P such that $L(\ell_{ij}) = i$ holds for all label points of the tessellation scaled by ω. While an exact computation for this value was not found for the unweighted two-dimensional case, an algorithm making use of the idea to approximate tessellations by Voronoi diagrams was proposed and applied to different experimental contexts. In this section, the findings shown in section V are discussed, as well as their implications for the algorithm's performance/correctness.

In the context of reconstructing generator points from randomly-generated Voronoi tessellations, the main challenge faced is the determination of the parameters ϕ_i, which control how much the generator points are nudged in each iteration of the algorithm. Figure 8 indicates that for the chosen experimental values, the median value for ω appears to lie around 0.95 for most values of ϕ, except for the values to the left-most and the right-most sides, where the median value of ω slightly decreases. The discrepancy between the input tessellation and the final approximation is generally a better indication of accuracy than ω alone, since ideally the area discrepancy between the input tessellation and the algorithm's output should be minimal. This idea behind this discrepancy measure is used as the primary performance measure by Suzuki & Iri [9]. Most values of ϕ indicate a median area discrepancy of $\sim 0.7\%$, except yet again for the tail ends of the test values which yield a higher median area discrepancy. The reason could be that, in the case where $\phi = 0.005$, the nudging step moves the generator points too slowly for the algorithm to yield meaningful results, leading in early termination. In the case where $\phi \geq 0.045$, the algorithm is more likely to overdo the nudging process, and thus misses correct positions for the generator points.

As for determining a good value for ϕ among the test values, Figure 9 shows the trajectory of the relationship between the median processing time in seconds and the median area discrepancy in terms of ϕ. Both $\phi = 0.02$ and $\phi = 0.025$ seem to perform the best among the tested values, as their results offer a good compromise between processing time and accuracy of the output. Because of this, $\phi = 0.02$ was chosen for the remaining experiments.

Figure 7 shows a best-case example of the algorithm's output, where the algorithm determined a lower bound of $\omega = 0.9999905$ for the input tessellation. However, there is still a considerable variance in the results yielded by the algorithm in this use case, warranting further research into additional factors that may impact the algorithm's application here, e.g., examining the impact of the input tessellation's structure and appropriately adapting ϕ.

In the next experiment, a hexagonal Voronoi tessellation, where the centroids of each region correspond to the tessellation's generator points, was iteratively distorted to determine

the effectiveness of the algorithm when tasked with approximating these tessellations. Recall that the algorithm initializes its guess for the generator points to be the centroids of every region. Figure 10 shows such a distorted tessellation, where the vertices have been distorted by a factor of 3%, as well as the approximation yielded for this tessellation with an area discrepancy of 9%. Figure 11 shows the box-and-whisker plots for ω and ϕ for each tested distortion factor. Since the distortion does not guarantee that the input tessellation retains its Voronoi properties, the more the vertices are distorted the less accurate the approximations become. The results indicate that, as expected, the closer the input tessellation is to being a proper uniformly-sized tiling, the better the resulting approximations. If an input tessellation resembles a centroidal Voronoi tessellation on visual inspection, then the algorithm may be worth applying here.

In both of the previous experiments, ω and δ appear inversely correlated, suggesting that the higher a tessellation's ω-measure is, the better it can be approximated by a Voronoi tessellation. The measure, if an exact computation were found, could be used to determine in advance if it would make sense to approximate a tessellation by a Voronoi tessellation.

Lastly, figure 12 shows this paper's approximation to figure 6 using $\phi = 0.02$. The algorithm took about 5 seconds to compute the solution with an area discrepancy of 3.08%, a marginal improvement of Suzuki & Iri's solution of 3.8%. While this result appears promising, further experimentation would warrant running Suzuki & Iri's algorithm on more tessellations, then comparing the results to those obtained by this paper's algorithm. This was not done in this research due to time constraints.

Ultimately, is the proposed ω-measure a reasonable measure for how well arbitrary tessellations conform to properties of Voronoi tessellations? The results indicate that the algorithm making use of the measure's theoretical framework performs reasonably well in most cases, albeit open to more fine-tuning and experimentation. Additionally, this paper fails to provide an exact computation for the ω-measure. Perhaps a better algorithm using the proposed framework could be created to yield a more exact version of the measure, as the algorithm proposed merely makes use of the presented ideas to create approximations rather than focusing on the ω-measure itself.

VII. Conclusion & Future Work

This paper defined a measure for Voronoi-likeness of arbitrary tessellations. The proposed ω-measure captures the distance properties of Voronoi tessellations, and the theoretical framework behind it was used to develop an algorithm to approximate arbitrary tessellations by Voronoi diagrams. A good value for the primary parameter of the algorithm, the nudging factor ϕ, was experimentally deduced to be 0.02 among other hand-picked test values. In the case of finding generator points of proper Voronoi tessellations, the algorithm yielded a median area discrepancy of $\sim 0.5\%$, and it was found that, due to the initial guess made by the approximation algorithm, tessellations with potential centroid generator points yield the best results for the algorithm. Additionally, a result from a previous paper was improved from an area discrepancy of 3.8% to one of 3.08%. The results appear promising, and the proposed ω-measure and its theoretical background serve as a proof of concept for how Voronoi-ness of tessellations can be expressed.

Future work entails further experimentation with the algorithm, especially in cases where the input tessellation does not correspond to a Voronoi tessellation. It is also necessary to look further into fine-tuning the parameters ϕ_i. Throughout this paper, a singular value was chosen for all ϕ_i through experimental means, but this process could be done more thoroughly. Perhaps ϕ_i could be fine-tuned based on the structure of the initial input tessellation, as it may be more wise to increase the nudging in certain regions than others.

References

[1] Narciso Aguilera, Belen Palop, and Hebert Pérez-Rosés. Approximating imprecise planar tesselations with voronoi diagrams. In *Proceedings of VISUAL 2016*, pages 19–23, 11 2016.

[2] Peter F. Ash and Ethan D. Bolker. Recognizing dirichlet tessellations. *Geometriae Dedicata*, 19:175–206, 1985.

[3] George W. Barlow. Hexagonal territories. *Animal Behaviour*, 22:876–IN1, 1974.

[4] Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, dec 1979.

[5] Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In *International Symposium Graph Drawing and Network Visualization*, 2004.

[6] Masami Hasegawa and Masaharu Tanemura. On the pattern of space division by territories. *Annals of the Institute of Statistical Mathematics*, 28:509–519, 02 1976.

[7] David Heath and Simon Kasif. The complexity of finding minimal voronoi covers with applications to machine learning. *Computational Geometry*, 3(5):289–305, 1993.

[8] Hisao Honda. Description of cellular patterns by dirichlet domains: The two-dimensional case. *Journal of Theoretical Biology*, 72(3):523–543, June 1978.

[9] Atsuo Suzuki and Masao Iri. Approximation of a tessellation of the plane by a voronoi diagram. *Journal of the Operations Research Society of Japan*, 29(1):69–97, 1986.

VIII. Appendix

A. Derivation for ω given P

Let $p = p_i, q = p_{k\neq i}, v = v_j$, and $\ell = \ell_{ij}$. Let $d(A, B)$ be the euclidean distance between two points A and B.

$$\begin{aligned}
d(p,\ell) &\leq d(q,\ell) \\
\sqrt{(p_x-\ell_x)^2+(p_y-\ell_y)^2} &\leq \sqrt{(q_x-\ell_x)^2+(q_y-\ell_y)^2} \\
(p_x-\ell_x)^2+(p_y-\ell_y)^2 &\leq (q_x-\ell_x)^2+(q_y-\ell_y)^2 \\
p_x^2-2p_x\ell_x+\cancel{\ell_x^2}+p_y^2-2p_y\ell_y+\cancel{\ell_y^2} &\leq q_x^2-2q_x\ell_x+\cancel{\ell_x^2}+q_y^2-2q_y\ell_y+\cancel{\ell_y^2} \\
-2p_x\ell_x-2p_y\ell_y+2q_x\ell_x+2q_y\ell_y &\leq q_x^2+q_y^2-p_x^2-p_y^2 \\
\ell_x(2q_x-2p_x)+\ell_1(2q_y-2p_y) &\leq q_x^2+q_y^2-p_x^2-p_y^2
\end{aligned}$$

Since $\ell_{ij} = p_i + \omega(\overrightarrow{p_i v_j})$, $\ell_x = p_x + \omega(v_x - p_x)$ and $\ell_y = p_y + \omega(v_y - p_y)$.

$$\begin{aligned}
(p_x+\omega(v_x-p_x))(2q_x-2p_x)+(p_y+\omega(v_y-p_y))(2q_y-2p_y) &\leq q_x^2+q_y^2-p_x^2-p_y^2 \\
\omega\left[(v_x-p_x)(2q_x-2p_x)+(v_y-p_y)(2q_y-2p_y)\right] &\leq q_x^2+q_y^2-p_x^2-p_y^2-2p_x(q_x-p_x)-2p_y(q_y-p_y)
\end{aligned}$$

Let $\Psi = [(v_x - p_x)(2q_x - 2p_x) + (v_y - p_y)(2q_y - 2p_y)]$, and let Θ be the right hand side of the inequality above. Then:

$$\omega \leq \begin{cases} \frac{\Theta}{\Psi} & \text{if } \Psi > 0 \\ -\frac{\Theta}{\Psi} & \text{if } \Psi < 0 \end{cases}$$

If $\Psi = 0$, then either $v_x = p_x$ and $p_y = q_y$, or $v_y = p_y$ and $p_x = q_x$.

Case 1 - $v_x = p_x$ and $p_y = q_y$:

$$\begin{aligned}
\omega\left[(p_x-p_x)(2q_x-2p_x)+(v_y-p_y)(2q_y-2p_y)\right] &\leq q_x^2+\cancel{q_y^2}-p_x^2-\cancel{q_y^2}-2p_x(q_x-p_x)-2p_y(q_y-q_y) \\
0 &\leq q_x^2-p_x^2-2p_x(q_x-p_x) \\
0 &\leq q_x^2-p_x^2-2p_xq_x+2p_x^2) \\
0 &\leq q_x^2-2p_xq_x+p_x^2 \\
0 &\leq (q_x-p_x)^2
\end{aligned}$$

This always holds, so the choice of ω irrelevant & unrestricted here. So choose $\omega = 1$ in this case.

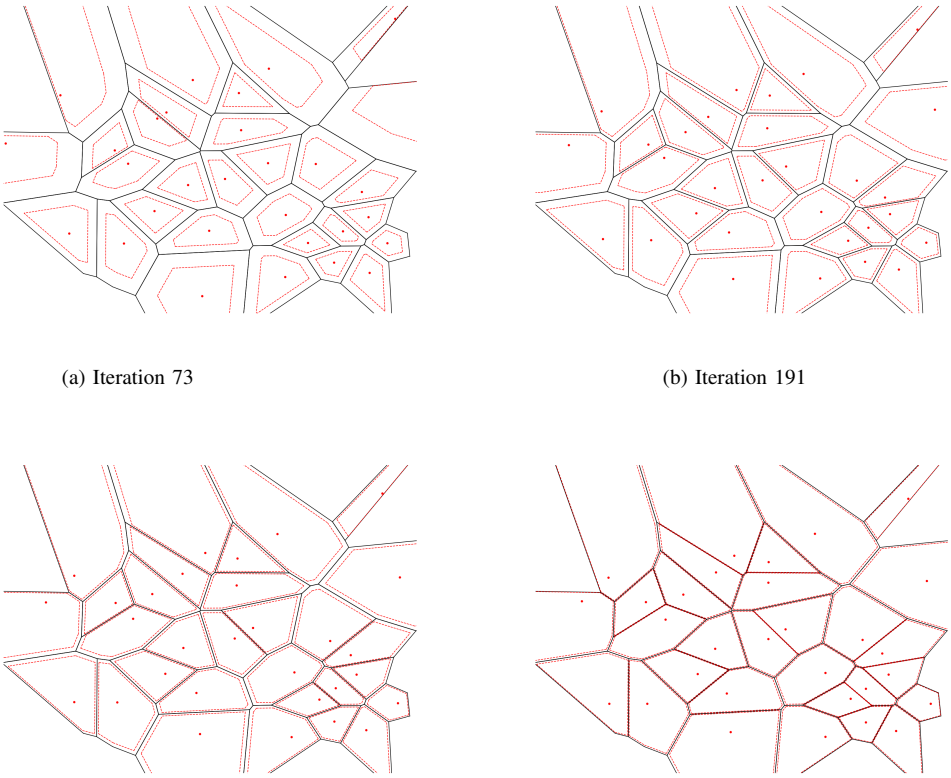
Case 2 - $v_y = p_y$ and $p_x = q_x$:

$$\begin{aligned}
\omega\left[(v_x-p_x)(2q_x-2q_x)+(p_y-p_y)(2q_y-2p_y)\right] &\leq \cancel{p_x^2}+q_y^2-\cancel{p_x^2}-p_y^2-2p_x(q_x-q_x)-2p_y(q_y-p_y) \\
0 &\leq q_y^2-p_y^2-2p_y(q_y-p_y) \\
0 &\leq q_y^2-p_y^2-2p_yq_y+2p_y^2 \\
0 &\leq q_y^2-2p_yq_y+p_y^2 \\
0 &\leq (p_y-q_y)^2
\end{aligned}$$

This also always holds, and the choice of ω is irrelevant yet again. So in either case, if $\Psi = 0$, choose $\omega = 1$ to obtain the following:

$$\omega \leq \begin{cases} \frac{\Theta}{\Psi} & \text{if } \Psi > 0 \\ 1 & \text{if } \Psi = 0 \\ -\frac{\Theta}{\Psi} & \text{if } \Psi < 0 \end{cases}$$

B. Growing islands



(a) Iteration 73

(b) Iteration 191

(c) Iteration 363

(d) Iteration 665

Fig. 13. Selected steps from Algorithm 2's process for the Inverse Voronoi problem using $\phi = 0.02$