ALGORITHM FOR COMPUTATION AND VISUALIZATION OF WEIGHTED CONSTRAINED VORONOI DIAGRAMS

A Thesis presented to the Faculty of the Graduate School
University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
CHAOWALIT THAMSONGLAR

Dr. Kannapan Palaniappan, Thesis Supervisor

DECEMBER 2009
The undersigned, approved by the Dean of the Graduate School, have examined the thesis entitled

ALGORITHM FOR COMPUTATION AND VISUALIZATION OF WEIGHTED CONSTRAINED VORONOI DIAGRAMS

presented by Chaowalit Thamsonglar

a candidate for the degree of Master of Science

and hereby certify that in their opinion it is worthy of acceptance.

_____________________________________________________
Dr. Kannapan Palaniappan

_____________________________________________________
Dr. Youssef Saab

_____________________________________________________
Dr. David R. Larsen
ACKNOWLEDGEMENTS

This work would be impossible without the following people Dr.Kannapan Palaniappan who is my advisor here at University of Missouri-Columbia and also Dr.David Larsen who supported me in all possible ways.

Additionally, without the support of my colleagues, Ian Scott and Eric McDavid, this thesis would have been much more difficult to overcome problems in this work.

I would also like to thank for support from my friend Kanchana Songkittiphong and Kittissak Sajjapongse for letting me discuss about this algorithm and giving me idea, sasiwimon yoo-eam and patsharaporn techasintana for spending time and thoughts trying to come up with biological application that this algorithm can be applied.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................................................................................. ii

LIST OF FIGURES ................................................................................................................... vi

LIST OF TABLES ...................................................................................................................... ix

ABSTRACT ............................................................................................................................... x

CHAPTER 1 INTRODUCTION ................................................................................................. 1

1.1 Introduction ....................................................................................................................... 1

1.2 Research objective ........................................................................................................... 3

CHAPTER 2 REVIEW OF LITERATURE AND RELATED TOPICS ................................. 4

2.1 Literature review .............................................................................................................. 4

2.2 Area Potentially Available (APA) ................................................................................ 7

2.3 Voronoi Growth Model (VGM) .................................................................................... 8

CHAPTER 3 THE WEIGHTED CONSTRAINED VORONOI DIAGRAMS .................. 11

3.1 Preliminaries .................................................................................................................... 11

3.2 Methodologies ............................................................................................................... 13

3.2.1 Find all neighbors .................................................................................................... 18

3.2.2 Compute oriented chords ....................................................................................... 19

3.2.3 Compute interior intersecting point .................................................................... 22
3.2.4 Validate boundary points and interior intersecting point ........................................ 24

3.5 Constructing final polygon ...................................................................................... 32

CHAPTER 4 EXPERIMENTAL RESULTS ........................................................................ 35

4.1 Comparing different Voronoi diagrams .................................................................... 36

4.2 Forestry result ........................................................................................................... 39

4.2.1 Plantation .............................................................................................................. 39

4.2.2 Voronoi models for characterizing natural forest stands ....................................... 40

4.3 Biomedical application of Voronoi diagrams .......................................................... 46

4.4 Complexity analysis ................................................................................................. 49

CHAPTER 5 SYLVAN-APA SOFTWARE TOOL FOR FORESTRY APPLICATIONS
........................................................................................................................................ 52

5.1 Sylvan-APA input ..................................................................................................... 52

5.2 Sylvan-APA display .................................................................................................. 53

5.3 Sylvan-APA output ................................................................................................... 55

CHAPTER 6 CONCLUSIONS AND FUTURE WORK ................................................... 56

6.1 Conclusions ............................................................................................................... 56

6.2 Future work ............................................................................................................... 57

REFERENCE .................................................................................................................. 59
APPENDIX A USER’S MANUAL

A.1 Introduction

A.2 Sylvan-APA main window

A.3 Sylvan-APA preference dialog

A.4 Menus, Toolbars and Status bar

A.4.1 Menus

A.4.2 Toolbars

A.4.3 Status bar

A.5 Keyboard Shortcuts

A.6 File formats
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Circle $C_1$ with and its neighbor</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Circle $C_1$ with the oriented chords created from its neighbor</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Estimated boundary for $C_1$</td>
<td>13</td>
</tr>
<tr>
<td>3.4 The weighted constrained Voronoi diagram system flowchart</td>
<td>17</td>
</tr>
<tr>
<td>3.5 Two circles intersection</td>
<td>19</td>
</tr>
<tr>
<td>3.6 The oriented chords $P_1P_2$ is counter-clockwise direction if $C_2P_1P_2$ form a clockwise direction</td>
<td>21</td>
</tr>
<tr>
<td>3.7 Two lines intersection</td>
<td>22</td>
</tr>
<tr>
<td>3.8 The pseudo lines are drawn from center of the circle to each boundary point and interior intersecting point</td>
<td>25</td>
</tr>
<tr>
<td>3.9 The generator $C_1$ center is occupied by another generator.</td>
<td>27</td>
</tr>
<tr>
<td>3.10 This is the case when covering circle method does not work correctly.</td>
<td>28</td>
</tr>
</tbody>
</table>
3.11 P_2 and P_3 should be removed because it forms a clockwise triangle with one of
the oriented chords ................................................................. 30

3.12 A special case that oriented half plane can work correctly but line crossing
method does not ................................................................. 31

3.13 A special case that oriented half plane work correctly but covering circle method
does not ................................................................. 31

3.14 Two generators C_2 and C_3 create the same oriented chords on C_1........... 33

3.15 This is the final result for C_1 with its validated points and direction........... 33

4.1 The standard Voronoi diagram on generated dataset. ...................... 37

4.2 The multiplicatively weighted Voronoi diagram on generated data....... 38

4.3 The weighted constrained Voronoi diagram on generated dataset. 38

4.4 Plantation dataset .................................................................. 39

4.5 The weighted constrained Voronoi diagram of plantation dataset........ 39

4.6 The original image of American sycamore (platanus occidentalis L.) standat
the Basket Wildlife Research Area near Ashland Missouri .................. 41

4.7 The original image with crown polygons and centers marked by an expert. 42

4.8 The weighted constrained Voronoi diagram calculated from center and radii
with tree crown drawn in red .................................................. 42

4.9 The original image of Bob Veech Road stand at the Basket Wildlife Research
Area near Ashland Missouri ..................................................... 43
4.10 The original image with crown polygons and centers marked by an expert. 43

4.11 The weighted constrained Voronoi diagram calculated from centers and radii with tree crown drawn in red. ................................................................. 44

4.12 The original image of Walnut Road stand, at the Basket Wildlife Research Area near Ashland Missouri. ........................................................................... 44

4.13 The original image with crown polygons and centers marked by an expert. 45

4.14 The weighted constrained Voronoi diagram calculated from centers and radii with tree crown drawn in red. ................................................................. 45

4.15 Histopathology data provided by Michael Feldman (Department of Surgical Pathology, University of Pennsylvania). ............................................... 46

4.16 Extracted data from histopathology data set. Black dot is a center of the cell and the red circle represent its boundary. ...................................................... 47

4.17 The weighted constrained Voronoi diagram on histopathology data 47

4.18 A mask of wound image with four unique foreground colors obtained from Nath et al. algorithm. ................................................................................. 48

4.19 An extracted data image from wound image. ......................................... 48

4.20 The weighted constrained Voronoi diagram on wound image data 49

5.1 An example of weighted constrained Voronoi diagram being drawn in Sylvan-APA .................................................................................................... 53

5.2 Sylvan-APA preference dialog. ......................................................... 54
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The overall computation time and number of hidden generators for different sizes of datasets on the same area of space</td>
</tr>
<tr>
<td>4.2</td>
<td>The table for time to search for overlapping generator, time to estimate new boundary, and estimated density on different sizes of datasets on the same area of space</td>
</tr>
</tbody>
</table>
ALGORITHM FOR COMPUTATION AND
VISUALIZATION OF WEIGHTED CONSTRAINED
VORONOI DIAGRAMS

Chaowalit Thamsonglar

Dr. Kannpan Palaniappan, Thesis Supervisor

ABSTRACT

A weighted Voronoi diagram (WVD) has been developed in order to regionalize or allocate space around a predetermined set of points or generators for the case that generator points have different weights reflecting their variable properties [16]. In this research, the weighted constrained Voronoi diagram has been adapted in order to fix some weaknesses in the older method. The new method allows some points on a space to be unclassified and some center points to be merged with other center points with higher weights to avoid incorrect spatial neighborhood graphs which lead to poor segmentation. Mainly, this algorithm generates the collection of geometric objects by using center points and their maximum radius given by users. Also, additional boundaries are created to determine classification of the overlapping area by using a linear equation. Finally, the obtained results show the visualization of the space partitioned by the collection of geometric objects that can be applied to diverse problem sets, including forestry spacing, cell biology, and the optimal allocation of resources.
CHAPTER 1

INTRODUCTION

1.1 Introduction

Computational geometry is an important science used in many applications such as computer graphics, computer-aided design, robotics, pattern recognition, and operations research, and research has been done since the mid-1970s. This field uses algorithms designed to analyze geometrical problems. In recent years, many interesting developments have been made in the field of computational geometry, such as the construct known as a Voronoi diagram. Voronoi Diagrams are a set of approaches to divide up the space in a field among a set of points such that the area inside the polygon is closest to the polygon’s centroid. Since the Voronoi diagram is one of the most fundamental data structures in computational geometry and the powerful tool in solving seemingly unrelated computational problems, it receives a lot of attention from researchers [4].

A Voronoi diagram is a popular method used to find the object that is closest to the given point. However, an ordinary Voronoi diagram assumes that every point in the space is occupied by one object or another, and that the center point of each object has the same weight [25]. Hence, the area of a point that is closest to the edges can be an unnecessarily large.

A weighted Voronoi diagram is an adaptation of a Voronoi diagram in which the weight of each center point may be specified in order to varying size of the objects at the
points. However, in the normal weighted Voronoi diagram, an object can never completely vanish. In practical applications, this is not always true. For an example, let assume that the market available to a location can be described as a circle, and its radius is the size of the marketplace. If a very small marketplace is very close to a very big marketplace, the small marketplace might have to permanently close. That means, it can be completely overwhelmed by the larger competitor. Because of this situation, the algorithm allows the point to vanish when it no longer has any area left in the space.

The weighted constrained Voronoi diagram (WCVD) allows users to specify the maximum radius of each center point. This is useful for problems where the extent of the influence of a center is limited by some theory. Moreover, this method allows one polygon to completely absorb another one. This method can be applied in many areas especially in forestry spacing applications. For forestry field, the forest floor is considered as Euclidean space, and center point of a tree is considered as a centroid of the Voronoi polygon. In this case, not every place in the space must be occupied by Voronoi polygons. One of the main problems that forest scientists have concerned is estimating areas potentially available (APA) which is used to define that space available to an individual tree competitive status or available growing space [32, 26]. This method can be used not only to partition the forest floor into areas potentially available to individual trees, but also to predict competition influences on individual tree growth, so the researchers can compute the growth rate and asymmetry in growth direction of the trees. This approach can be used to present how crowded of each tree is in the forest, and predicting that which trees are most likely to die.
1.2 Research objective

In forestry, the weighted constrained Voronoi diagram can be used to determine Area Potentially Available (APA). The objectives of this thesis are (1) to develop a weighted constrained Voronoi diagram that is based on APA methodology, and (2) to develop an easy to use program for foresters to calculate these complex spatial patterns. This software needs to be intuitive enough to allow users easily understand how to use the software, and (3) to determine advantages and disadvantages of the weighted constrained Voronoi diagram by comparing this method with other common Voronoi diagrams.

In chapter 2, we will review the literature on Voronoi diagrams and present discussion on the use of Voronoi model for tiling the space. We will explain the APA approach. In chapter 3, the weighted constrained Voronoi developed in this research is described in details with examples. In chapter 4, the model is used to apply to various applications, and results are analyzed. In chapter 5, a Sylvan-APA program is presented, which is a forestry application to allocate space in a forest base on the weighted constrained Voronoi method. In chapter 6, the conclusions of this study to develop the software are presented along with suggestions for future direction of this research work.
CHAPTER 2

REVIEW OF LITERATURE AND RELATED TOPICS

2.1 Literature review

Many space segmentation approaches rely upon or are enhanced by using spatial relationship information between sub-regions and their object correspondences. Spatial analyses of this problem set in many applications have been solved by a Voronoi diagram technique [20]. By using a set of generating points, a Voronoi diagram simply segments a plane into regions by comparing the Euclidean distance from each point to each generating point, and each point will belong to the region of the closest generating point.

The Voronoi diagram has been studied since the early-1900s, the first papers were written by Georgy Fedoseevich Voronoi and the early articles were based on graphing the space on paper. However, efficient algorithms did not appear until the 1970s, with the birth of the field of computational geometry and the development of available computers. The Voronoi diagram has been used in various applications in areas such as computer graphics, geophysics, and meteorology [19, 33]. More recently, Voronoi diagrams have been useful in a number of computer graphics related applications, including shape analysis [12], surface reconstruction [18, 23], three dimensional interpolation [21], skeleton extraction [16], collision detection [11], and map algebra framework [22]. Moreover, Voronoi diagram has been adapted in various ways to specialized applications.
such as weighted Voronoi diagram, hierarchic Voronoi diagram [1], and Kinetic Voronoi diagram [17].

In order to make the Voronoi diagrams more applicable to some other practical applications, such as the population of a city and the level of risk of accident at a given point, weights are taken into account. In this case, each point in the plane still belong to the region of the generating point that provide the closest distance, but distances are computed in various ways depending on the weights. The various methods can be either multiplicatively, additively, or compoundly weighted Voronoi diagrams [27]. For a multiplicatively weighted Voronoi diagram (MWVD), the distance is equal to the Euclidean distance divided by the weight of that generating point [28]. For an additively weighted Voronoi diagram (AWVD), the distance is equal to the Euclidean distance minus the weight of that generating point [31]. For a compoundly weighted Voronoi diagram (CWVD), the distance is equal to Euclidean distance minus the first weight and then divided by the second weight of that generating point [14]. Since there are many articles related to the weighted Voronoi diagram method and here are some of the more relevant examples.

In 2004, Reitsma, Sethia, and Trubin [28] adapted MWVD to regionalize and visualize regions of information space. They inversed the traditional MWVD to generate Voronoi regions whose relative area sizes are known in order to find a set of growth rates rather than resultant area. By using an adaptive, iterative approach, to adjust the weight set based on the error of the previous solution, they overcome the main disadvantage of the MWVD.
Curci, Galvão, Novaes, and Souza [13] also applied MWVD to logistics distribution problems in 2004. The model was applied to solve a parcel delivery problem in the city of São Paulo, Brazil by smoothing district contours to close to the configuration contours encountered in practical situation. The resulting repartition of the region led to more balanced time/capacity utilization (load factors) across the districts.

In 2007, Sharifzadeh and Shahabi [31] applied AWVD to the problem space of the Optimal Sequenced Route (OSR) query in vector and metric spaces. In order to find a route of minimum length starting from a given source location and passing through a number of typed locations in a specific sequence imposed on the types of the locations, they exploited the geometric properties of the solution space and identified the AWVD as its corresponding geometric representation.

Alliez, Antani, and Delage [3] also constructed AWVD to generate mesh sizing function in 2007. By incorporating the AWVD into an algorithm for computing mesh sizing functions, they avoided iterating over every input point at each point query. The resulting implementation was much faster than previous methods in practice, and the sizing field computed by their method was not an approximation so they did not end up sacrificing accuracy for speed.

In 2008, Cursi, Gonçalves, Novaes, and Oliveira [14] developed a computational tool based on the concepts of CWVD and Non-linear Programming to study and define the ideal bus-stop spacing in urban areas in order to minimize the total travel time of the passengers of a bus line. To find the optimal solution, the model used the density function
of the distribution of the population in the affected area and combined it with the model of the CWVD to search for the minimum value using the usual methods of non-linear programming.

2.2 Area Potentially Available (APA)

This term originally comes from forestry in 1965 [8]. Area potentially available has been known by forester as the way to compute competition influences on individual tree. However, APA has only once been available in computer software (Fortran from the mid-1980’s). No modern computer software is available. In the past, by drafting, foresters used APA to evaluate the stand density applicable to a given tree in a stand. They assumed that in a mapped stand, each tree had potentially available to it half the distance to the neighboring tree, everything else being equal. Also, a line drawn at right angles to and bisecting the line joining the trees would define the available space boundary between these two, and a series of lines similarly drawn around any tree would demarcate the area potentially available to that tree. This assumption was matched to standard Voronoi diagram.

In 1981, Daniels created a computation of the APA index called program COMPAPA by using a FORTRAN computer program [26]. Firstly, this program initiates a search for influential competitors around each subject tree. Secondly, it assembles a prospective list of those trees which could influence the construction of the subject tree’s polygon. Thirdly, it computes all possible intersections of each competitor’s influence lines with one another. Then it constructs the polygon using a minimum subset of
competitors. However, corresponding to the fact that the characteristics of two trees can seldom be so precisely equal, Daniels and Nance used relative basal area as the weighting factor to locate the influence between the subject tree and each competitor in 1983 [26]. Also, they created an index of relative influence in order to provide additional information about the relative influence of a competitor with respect to the other competitors used in constructing a subject tree’s polygon. This concept is also matched to weighted Voronoi diagram.

2.3 Voronoi Growth Model (VGM)

In many fields such as material science, and plant ecology, Voronoi diagram has been adapted to predict the pattern or shape of particle statistics or some materials such crystal and trees, named Voronoi growth model (VGM). A crystal growth model can be used as an example to explain how the VGM works. VGM can be a planar segmentation diagram for crystal growth. If all crystals start growing simultaneously at the same speeds, standard Voronoi diagram will be used [24]. However, in the case that each crystal start growing at different time with the same speed, AWVD will be taken into account [2], and if crystals grow at different speeds in the same time, MWVD will be concerned [24]. Finally, if each crystal starts growing at different time with different speed, CWVD will be used. Here are some articles related to Voronoi growth model.

In 1992, Schaudt proposed the multiplicatively weighted crystal growth Voronoi diagram that can be used to model crystal growth when the crystals have different constant growth rates [29]. For this model, the distance from a site to a point in its region
is measured along a shortest path lying entirely within the region, and a growing crystal (or region) may “wrap around” another site’s region. Concerning about time consuming during calculation process, this model used a convex polygon distance function in order to construct the diagram that the worst case complexity of the diagram is $O(n^2)$ even though there are $n$ regions.

In 2000, Mioc, Anton, and Gold presented a method for the visualization of the nucleation and growth of particles based on an algorithm for the dynamic construction of additively Voronoi diagram [2]. They used the Poisson point process in the dynamic additively Voronoi diagram to generate the Johnson-Mehl tessellation. The Johnson-Mehl model is a Poisson Voronoi growth model, in which nuclei are generated asynchronously using a Poisson point process, and grow at the same radial speed. These algorithms would be useful in the visualization of the spatial growth processes investigated in material science and in the determination of physical properties of materials.

In conclusion, Voronoi diagram can be used in not only computer graphics, geophysics, and meteorology, but also forestry application especially in an APA problem. In order to solve the APA problem, the weighted constrained Voronoi diagram (WCVD) is developed based on the concepts of Voronoi growth model (VGM). By adding some extra constraints to a standard VGM, WCVD can be solved some problem that VGM cannot. For example, in VGM, the area of the generator that has more weights than the one that close to it may wrap around the smaller area. In that case, it is not true for trees because a tree cannot grow and wrap around another tree. By allowing some generators
to be merged with other center points with higher weights, WCVD are more capable to be used to APA problem.
CHAPTER 3
THE WEIGHTED CONSTRAINED VORONOI DIAGRAMS

As discussed in the introduction and literature review, the weighted constrained Voronoi diagram has properties that differ from standard Voronoi diagrams and weighted Voronoi diagram and it can be used to solve some problems that the standard Voronoi diagram and the weighted Voronoi diagrams cannot.

3.1 Preliminaries

This algorithm estimates new boundary for any generator (generally circles) in the space. The generator in the space is represented with two parameters: center of the generator and weight. Weight specifies how far one generator can occupy the space. Hence, weight can also be presented as a radius. When any pair of generator overlaps, space needs to be allocated between these two generators. Figure 3.1 illustrates the generator C₁ with other four generators that overlap with C₁. The intersecting points between any pair of intersecting generators are then computed and the line between this pair of generator is used as a boundary line. In the complete version of this algorithm, the direction of this boundary line needs to be determined. Hence, let us call this boundary line an oriented chord. The oriented chord can also be referred to as half-edges. One oriented chord consists of two points. Let us call these two points boundary points since they are always located on the boundary of the generator. Figure 3.2 illustrates how oriented chords are used to divide the space for the generator C₁ and also how one
An oriented chord consists of two boundary points. The boundary points in Figure 3.2 are shown as yellow dots.

However, boundary points are not the only point that will be used to estimate new boundary in the weighted constrained Voronoi diagram. There can also be an interior point which is created when two oriented chords intersect within the generator. The purple points in Figure 3.2 are the interior intersecting points. Since the interior intersecting point is created when two oriented chords intersect, it is always located within the generator.

Figure 3.3 is an example of the final polygon of generator $C_1$ after validating and removing points that are not part of a new boundary. The final result can be considered as
a polygon in which edges can be either arcs or straight lines. Thus, each boundary point and interior intersecting point needs to have one extra attribute to indicate whether the line that leaves the point is an arc or a straight line. This can be done in either clockwise or counter clockwise direction. In this thesis, a counter clockwise direction was used. We will further discuss how to determine what type of line that leaves boundary point and interior intersecting point in its own section.

In conclusion, the final polygon for each generator consists of a set of boundary point and interior intersecting points with the extra attribute indicating if the line that leaves the point is an arc or a straight line.

3.2 Methodologies

To estimate new boundaries for each generator within the space, the weighted constrained Voronoi diagram has five main processes which are finding all neighbors,
computing the oriented chords and its direction, computing the interior intersecting point, validating the boundary and interior intersection points, and constructing the final polygon. The Figure 3.4 is the system flow diagram that is used by the weighted constrained Voronoi diagram.

**Algorithm 1 EstimateNewBoundary**

// This algorithm estimate new boundary for the generator C

**Input:** C, The generator

S, The set of all other generator in the space

**Output:** B, a final sorted polygon vertex list which represent new boundary of C

1: initialize K as an empty list of overlapping circles’ generator of C

2: initialize L as an empty list of oriented chords of C

3: initialize T as an empty list of interior intersecting points of C

4: initialize V as an empty list of validated points of C

5: //Finding overlapping circles

6: For each I ∈ S do

7: If neighbor(C,I) then

8: Append I to K

9: End if

10: End for
11: //Computing oriented chords

12: For each $I \in K$ do

13:  For each $J \in K$ do

14:   If ($I \neq J$) then

15:    Append computeOrientedChord($I,J$) to $L$

16:   End if

17:  End for

18: End for

19: //Computing interior intersecting points

20: For each $I \in L$ do

21:  For each $K \in L$ do

22:   If ($I \neq K$) then

23:    Append computeInteriorPoint($I,K$) to $T$

24:   End if

25:  End for

26: End for

27: //Validating interior intersecting point and boundary point

28: For each $I \in T$ do
29: If orientedHalfPlane(I,L) then
30: Append I to V
31: End if
32: End for
33: For each I in L do
34: Let t be a point at the tail of I
35: Let h be a point at the head of I
36: If orientedHalfPlane(t,L) then
37: Append t to V
38: End if
39: If orientedHalfPlane(h,L) then
40: Append h to V
41: End if
42: End for
43: //sort validated points and remove overlapping point
44: B = constructFinalPolygon(V)
Given a set of generators

Find all neighbors

Compute oriented chords

Compute interior intersecting points

Validate boundary points and interior intersecting points

- Line crossing
- Covering circle
- Oriented half-plane

Construct final polygon

Final polygon

Figure 3.4: The weighted constrained Voronoi diagram system flowchart.
3.2.1 Find all neighbors

In Figure 3.1, generator $C_1$ has four neighbors but it is possible that there can be many other generators in the space that do not intersect with $C_1$. To reduce further computational demand, a small computationally efficient method is used to determine if two particular generators are neighbor. Let $C_1$ and $C_2$ be generators with radii of $r_1$ and $r_2$ consequently. $C_1$ and $C_2$ are neighbors if Euclidean distance of $C_1$ and $C_2$ is equal to less than $r_1+r_2$.

Algorithm 2 Neighbor

Input: $C_1$, first generator

$C_2$, second generator

Output: Return true if $C_1$ and $C_2$ overlapped

1: Compute $d$ = Euclidean distance of $C_1$ and $C_2$

2: if $d$ is lesser than $r_1+r_2$ then

3: return true

4: else

5: return false

6: end if
3.2.2 Compute oriented chords

In this section, we will explain the method that is used to compute the oriented chords and establish the direction of the oriented chords. As mentioned in preliminaries, the oriented chords are used as a boundary line between two intersecting generators and the oriented chords needs to be established in order to determine the type of line that leaves from each boundary points in a counter-clockwise direction.

**Compute oriented chords**

To compute the oriented chords of two generators, the intersection of Two Generators method written by Paul Bourke is used [6].

![Diagram of two circles intersection](image)

Figure 3.5: Two circles intersection.

Considering the two triangles $C_1P_0P_1$ and $C_2P_0P_1$

\[
\begin{align*}
\text{1st circle:} & \quad a^2 + h^2 = r_1^2 \\
\text{2nd circle:} & \quad b^2 + h^2 = r_2^2
\end{align*}
\]

\[a \text{ can be solved using } d = a + b\]
\[ a = \frac{(r_1^2 - r_2^2 + d^2)}{2d} \] (3)

Then, \( h \) can be solved by substituting \( a \) in equation (1). \( h^2 = r_1^2 - a^2 \)

Since \( h \) is perpendicular to \( a \),

\[ X_1 = X_0 \pm \frac{h(Y_2 - Y_1)}{d} \] (4)

\[ Y_1 = Y_0 \pm \frac{h(X_2 - X_1)}{d} \] (5)

\( P_1 \) in term of \( X_1 \) and \( Y_1 \) are two boundary points that are part of the oriented chords. After the oriented chords position is obtained, the direction of the oriented chords will be then computed.

**Compute oriented chords direction**

If all oriented chords are established in the same direction which is counterclockwise in this thesis, one can then indicates the type of line that leaves from each boundary point. The line that leaves the boundary point at the tail of the oriented chords is a straight line and the line that leaves the boundary point at the head of the oriented chords is an arc. In Figure 3.6, the red line is the straight line that leaves from the tail of the oriented chords \( P_1 \) and the blue line is the arc that leaves from \( P_2 \) which is the head of the oriented chords.

From Figure 3.6, the oriented chords for \( C_1 \) is counterclockwise if and only if \( \overline{C_2P_1P_2} \) is clockwise. Robert Sedgewick’s CCW method [30] is then used to determine if \( \overline{C_2P_1P_2} \) is clockwise or not. By using right hand rule,

If \( \overline{C_2P_1} \times \overline{C_2P_2} > 0 \), \( P_1P_2 \) is counterclockwise
If $\overrightarrow{C_2P_1} \times \overrightarrow{C_2P_2} < 0$, $\overrightarrow{P_1P_2}$ is clockwise.

Figure 3.6: The oriented chord $\overrightarrow{P_1P_2}$ is counter-clockwise direction if $\overrightarrow{C_2P_1P_2}$ form a clockwise direction.

Finally, the oriented chord with its direction is obtained. The oriented chord contains two boundary points which will be further validated if they are parts of new estimated boundary. The line that leaves boundary point at the head of the oriented chords is an arc and it is a straight line for the boundary point at the tail of the oriented chords. However, if we make the oriented chords clockwise rather than counter-clockwise, the line that leaves the boundary point at the head will be an arc and the line that leaves the boundary point at the tail will be a straight line.

Algorithm 3 ComputeOrientedChord

Input: $C_1$, first generator

$C_2$, second generator

Output: l, oriented chord that divines the space between $C_1$ and $C_2$

1: Compute $d =$ Euclidean distance of center of $C_1$ and center of $C_2$
2: Compute a from equation 3

3: Compute h by substitute a in equation 1

4: Substitute h in equation 4 and 5 to obtain the two intersecting points $P_1$ and $P_2$

5: if $P_1P_2$ from counter clockwise triangle with $C_2$ then

6: \hspace{1cm} \textbf{return} \ P_2P_1

7: else

8: \hspace{1cm} \textbf{return} \ P_1P_2

9: \hspace{1cm} \textbf{end if}

---

### 3.2.3 Compute interior intersecting point

Each oriented chords within the generator can intersect with other oriented chords and create an additional point that can be part of new boundary for the generator. Since this intersection point is created from two straight lines, the line that leaves this point is always a straight line. To compute the interior intersection point, the Intersection Point of Two Lines by Paul Bourke is used [5].

![Figure 3.7: Two lines intersection.](image-url)
Two equations of the lines are

For line a
\[ P = P_1 + u_a(P_2 - P_1); \]  
\[(5)\]

For line b
\[ P = P_3 + u_b(P_4 - P_3); \]  
\[(6)\]

Equation (5) = (6)
\[ P_1 + u_a(P_2 - P_1) = P_3 + u_b(P_4 - P_3) \]  
\[(7)\]

Split point in equation (7) to x and y, we will obtain two equations

\[ X_1 + u_a(X_2 - X_1) = X_3 + u_b(X_4 - X_3) \]  
\[(8)\]

\[ Y_1 + u_a(Y_2 - Y_1) = Y_3 + u_b(Y_4 - Y_3) \]  
\[(9)\]

Solve equation (8) and (9) for \( u_a \) and \( u_b \)

\[ u_a = \frac{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)}{(y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3)} \]  
\[(10)\]

\[ u_b = \frac{(x_4 - x_3)(y_3 - y_1) - (y_4 - y_3)(x_3 - x_1)}{(y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3)} \]  
\[(11)\]

Substitute \( u_a \) in equation (5) or \( u_b \) in equation (6) to get the intersection point. For example, the intersection point (x, y) is

\[ X = X_1 + u_a(X_2 - X_1) \]  
\[(12)\]

\[ Y = Y_1 + u_a(Y_2 - Y_1) \]  
\[(13)\]

However, the intersection point needs to be tested to determine whether it lies within the range of these two finite lines. The intersection point will lie within the range of two lines if and only if \( u_a \) and \( u_b \) are greater than 0 but less than 1.
Algorithm 4 ComputeInteriorPoint

**Input:** $l_1$, first oriented chord

$l_2$, second oriented chord

**Output:** Return the intersecting point if two oriented chord intersect. Otherwise, return empty point.

1: From equation 10, compute the denominator of $u_a$

2: If the denominator $u_a \neq 0$ then

3: Compute $u_a$

4: Compute intersecting point with equation 12 and 13

5: **Return** intersecting point

6: Else

7: **Return** empty point because $l_1$ and $l_2$ do not intersect

8: End if

3.2.4 Validate boundary points and interior intersecting point

To validate boundary points and interior intersecting points we need to determine whether they should be part of new boundary for the generator. Three methods have been developed: line crossing, covering circle, and oriented half plane methods. However, the line crossing and covering circle methods have some limitations. There are some special cases in which both of these methods will not validate the boundary point and interior
intersecting point correctly as intended. Their special cases will be discussed in greater
detail later. On the other hand, the third method, oriented half plane can validate both
boundary points and interior intersecting points correctly.

**Line crossing**

This is the first validation method that was developed. The idea is that we are
trying to find the point closest to the center. Hence, the pseudo line is drawn from the
center of the generator to each point in consideration. If this pseudo line intersects any
other oriented chords, the point being validated should be removed. To determine
whether two lines intersect, the method discussed in chapter 3.2.3 is used.

For instance, in Figure 3.8, P₂ and P₃ should be removed because the line that is
drawn from C₁ to P₂ and C₁ to P₃ intersect with another oriented chords.

![Figure 3.8: The pseudo lines are drawn from center of the circle to each boundary point and interior intersecting point.](image)
Algorithm 5 LineCrossing

**Input:** $C_1$, the generator

- $P$, a point to be validated
- $l$, a list of oriented chord within the generator $C_1$

**Output:** Return true if $P$ will be a part of new boundary of $C_1$, otherwise return false.

1: **For** each $l$ do

2: **If** a pseudo line drawn from center of $C_1$ to $P$ intersect with $l$ **then**

3: **Return** false

4: **End** if

5: **End** for

6: **Return** true

However, this line crossing method does not validate correctly when center of the area at the center of the generator is occupied by another generator. In Figure 3.9, $P_1$, $P_3$ and $P_5$ should be parts of new boundary but line crossing method will validate that $P_1$, $P_2$ and $P_4$ are parts of new boundary.
Figure 3.9: The generator $C_1$ center is occupied by another generator.

**Covering circle**

This method derives from neighborhood list that enclose the subject point and the list of generator that originate the point to be validated. If the list of generators that enclose the point and the list of generators that originate the point are not equal, the point will be removed or will not be part of new boundary.

Let $P$ be a boundary point or an interior intersecting point. $G_P$ is the set of generator that originates $P$ and $L_P$ is set of generators which enclose $P$. $G_P$ can be defined by two conditions. If $P$ is a boundary point, member of $G_P$ consists of two intersecting generators. However, if $P$ is an interior intersecting point, $G_P$ is the combined list of
generators of two oriented chords that create this interior intersecting point. Point P should be removed if set $G_P$ is not equal to $L_P$.

For example, in Figure 3.8, generator $C_1$ has 4 boundary points and one interior intersecting point. The list of generator that originates $P_4$ and $P_3$ is $\{C_1, C_2\}$. The boundary point $P_3$ should be removed because the list of generator that encloses $P_3$ is $\{C_1, C_2, C_3\}$. However, boundary point $P_4$ should be kept since the list of generator that encloses it is $\{C_1, C_2\}$ and it is equal to the list of generator that originates $P_4$. As for the interior intersecting point $P_5$, the list of generators that originate it is $\{C_1, C_2, C_3\}$ and the list of generators that enclose is also $\{C_1, C_2, C_3\}$. Hence, the interior intersecting point $P_5$ should be kept.

![Figure 3.10: This is the case when covering circle method does not work correctly.](image-url)
**Algorithm 6** CoveringCircle

**Input:** P, a point to be validated

- o, a list of generator that originates P
- e, a list of generator that enclose P

**Output:** Return true if P will be a part of new boundary of C₁, otherwise return false.

1: If ( sizeOfList(o) != sizeOfList(e) ) then
2: Return false
3: End if
4: Return true

However, using this method, interior intersecting points always have three generators that originate it and boundary points always have two generators. This method does not work correctly when four or more generators intersect with each other because some of the interior intersecting point that needs to be part of new boundary gets removed. Figure 3.10 illustrates when this method does not work correctly. P₇ and P₈ are interior intersecting points and enclosed with all four generators. With covering circle method, P₇ and P₈ should not be parts of new boundary but in fact, P₇ and P₈ are parts of new boundary.

**Oriented half plane**

In this method, all intersection lines within C₁ need to be established in direction as discussed in section 3.2.2. Since the oriented chords are established in counter-
clockwise direction, any boundary point or interior intersecting point that forms a clockwise triangle to any oriented chords should be removed. Again, to determine if the boundary point or interior intersecting point forms a clockwise or counter-clockwise triangle, the method discussed in section 3.2.2 is used.

For instance, $P_2$ and $P_3$ in Figure 3.11 should be removed because $P_2$ forms a clockwise triangle with $P_3P_4$ and $P_3$ forms a clockwise with $P_1P_2$.

Figure 3.11: $P_2$ and $P_3$ should be removed because it forms a clockwise triangle with one of the oriented chords.

This method validates all the point correctly as we intended. It can handle the case in which line crossing and covering circle method cannot. Figure 3.12 and 3.13 illustrate Figure 3.9 and 3.10 after having computed the direction of the oriented chords of $C_1$. Figure 3.9 and Figure 3.10 are the cases when our previously developed method do not work correctly. In Figure 3.12, $P_3$ and $P_2$ will be removed because $P_3$ forms a clockwise
triangle with oriented chord $\overline{P_2P_1}$ and $P_2$ with $\overline{P_3P_4}$. As for Figure 3.13, this method will validate that $P_1$, $P_4$, $P_7$ and $P_8$ will be parts of new boundary for $C_1$ which is correct.

Figure 3.12: A special case that oriented half plane can work correctly but line crossing method does not.

Figure 3.13: A special case that oriented half plane work correctly but covering circle method does not.
**Algorithm 7 orientedHalfPlane**

**Input:** P, a point to be validated

I, a set of oriented chord within a generator C₁

**Output:** Return true if P will be a part of new boundary of C₁, otherwise return false.

1: **For** each oriented chord in the generator l **do**

2: **If** P forms clockwise triangle with l **then**

3: **Return** false

4: **End if**

5: **End for**

6: **Return** true

---

### 3.5 Constructing final polygon

At this point, those boundary points and interior intersecting points that are not part of new boundary should have been removed. All validated points need to be then sorted relatively to the generator center and it can be done either clockwise or counterclockwise. However, it has to be the same direction as oriented chord which is counterclockwise in this thesis. After having all the points sorted relatively to the generator center, we also need to remove a redundant point. Figure 3.14 illustrates how there can be redundant points within the same generator. $\overline{P₁P₂}$ is and oriented chord of C₁ that is created from an overlapping with C₂ and $\overline{P₃P₄}$ is created from an overlapping of C₁ to C₃.
However, those two oriented chords are exactly same lines and with the current method of drawing, this line will be drawn twice and it is unnecessary. We can simply remove this redundant point by checking if any pair of this validated point angle to the center is lesser than the threshold, then we remove one of them.

![Diagram](image)

Figure 3.14: Two generators $C_2$ and $C_3$ create the same oriented chords on $C_1$.

For instance, the result of sorting validated boundary point and interior intersecting point for Figure 3.15 should be $P_4P_5P_6P_1P_7$ and as discussed each point has one extra attribute indicates what type of line leaves the point. For example, the type of line that leaves $P_5$, $P_1$ and $P_7$ are straight line and $P_4$, $P_6$ are arcs.

![Diagram](image)

Figure 3.15: This is the final result for $C_1$ with its validated points and direction.
In conclusion, if the generator is overlapped with other generators, the space is divided up between overlapping generators and the new estimated boundary for each generator can be considered as a polygon which edges can be either arcs or straight lines. However, if the generators do not intersect with any other generators, their areas remain circles since they do not have to share the space with any other generators.

Algorithm 8 constructFinalPolygon

// This function sorts validated points and remove overlapping points out of the list

1: **Input:** C₁, the generator

2: \[ P, \text{a list of validated points} \]

3: **Output:** B, a final sorted polygon vertex list

4: \[ B = \text{sort}( \ P \ ) \] //Sort P relatively to the center of generator C₁

5: For \( i = 0; i < \text{sizeOfList}(B); i = i+1 \)

6: \[ \text{If} \ ( \| B_{(i+1)} - B_i \| < \delta \) \text{ then} \]

7: \[ \text{Remove} \ B_i \]

8: \[ \text{End if} \]

9: \[ \text{End for} \]
CHAPTER 4

EXPERIMENTAL RESULTS

This chapter deals with the usefulness of this weighted constrained Voronoi and presents results obtained through the use of this algorithm to the end. This chapter is divided into four sections. First, the results from the weighted constrained Voronoi diagram, the standard Voronoi diagram and, the multiplicatively weighted Voronoi diagram are presented to demonstrate the different space allocation from the different algorithms and to show that the weighted constrained Voronoi diagram partitions the space as expected. The second section evaluates the usefulness of the algorithm by investigating whether its space allocation of the overhead tree canopy area is representative of actual space allocation in a forest of trees. This is accomplished through a comparison of the space allocation by the algorithm with that obtained experimental data. The next section deals with the applications of the algorithm to other fields of research. The final section discusses the run-time complexity of the algorithm.

As mentioned in previous chapters, this algorithm can be used to allocate the space of any non mixing object that can be modeled as a circle. In the biological field, there are many objects such as cells, or tumor cells, plants, and fungus, which can be modeled as a circle. Hence, this algorithm is expected to be widely applicable in biological field as well. There may be other fields where this algorithm can be applied but in this thesis, only results from forestry and biology will be presented.
4.1 Comparing different Voronoi diagrams

In this section, we would like to compare the results from different space allocation algorithms which are standard Voronoi diagram, multiplicatively weighted Voronoi diagram and the weighted constrained Voronoi diagram. We will use a single of generator locations. However, for the weighted constrained Voronoi diagram and multiplicatively weighted Voronoi diagram, weights for each generator will be randomly assigned.

Figure 4.1 illustrates the standard Voronoi diagram and some properties that the weighted constrained Voronoi diagram fixes or changes. First, the generators around the edge tend to have more space than those in the middle. Second, the space is allocated with an equal division. Third, every point on the space must be occupied by one or another generator.

Figure 4.2 illustrates multiplicatively weighted Voronoi diagram and one of the properties of multiplicatively weighted Voronoi diagram that is different to standard Voronoi diagram is that the space is not allocated with an equal division but rather with the weights that were specified.

Lastly, Figure 4.3 illustrates weighted constrained Voronoi diagram on the same dataset. Figure 4.3 shows that all the points on the space do not have to be occupied, the space is allocated with the weights that were specified and the generators around the edges are not unnecessarily large. Also, the weighted constrained Voronoi diagram has another property that standard Voronoi diagram and weighted Voronoi diagram do not.
The weighted constrained Voronoi diagram allows the generator to vanish if all the space it has is occupied by other generators and the red circle in Figure 4.3 illustrates this.

Figure 4.1: The standard Voronoi diagram on generated dataset.
Figure 4.2: The multiplicatively weighted Voronoi diagram on generated data.

Figure 4.3: The weighted constrained Voronoi diagram on generated dataset.
4.2 Forestry result

After having shown the different properties of the weighted constrained Voronoi diagram, standard Voronoi diagram, and standard weighted Voronoi diagram. We now would like to show how the weighted constrained Voronoi diagram can be used in forestry. As far as the state of art in forestry is at the moment, the weighted constrained Voronoi diagram is mainly used to represent trees and trees can be classified into two types. First is the one that naturally grow and the second one is planted by human. Below, we will present the result from both plantation dataset and naturally grown forest. On naturally grown forest, we will also illustrate the quality of this space allocation for trees.

4.2.1 Plantation

Figure 4.4 and Figure 4.5 show results of the weighted constrained Voronoi diagram on plantation dataset and its result.

![Figure 4.4: Plantation dataset.](image)

![Figure 4.5: The weighted constrained Voronoi diagram of plantation dataset.](image)
4.2.2 Voronoi models for characterizing natural forest stands

We have three datasets for natural growth forest which are at the Baskett Wildlife Research Area, The School of Natural Resources, and University of Missouri, located east of Ashland Missouri. The samples consists of a young American sycamore (Platanus occidentalis L ) (Baskett Sycamore), a mix species stand along Bob Veech road at the Basket Wildlife Research area (Baskett Bob Veech), an old mixed species stand on Walnut Road at the Baskett Wildlife Research Area (Baskett Walnut). All stand we photographed from the above at an elevation of about 150 feet using balloon photography, and crown and the polygon centers area manually outlined. The tree centers were used as the generator location an assumed large radius was applied to each tree to evaluate the space allocation algorithm and its relation to tree space unitization. The results from the weighted constrained Voronoi diagram and the tree crown that are manually outlined are compared.

The results show that the polygon agreement is better for the smaller crown of tree. Baskett Sycamore stand has the most satisfactory result because all the lines that divide the space between trees are closest to the crowns that are manually outlined. This is because circle represents smaller tree better. When tree grows, its shape is from by its environment and it tends to grow more in a certain direction which has more growing space (light, water and nutrition) that are needed by the tree. In Basket Walnut stand, there are several trees with the big crowns and it is apparent that several trees were in the space but have died leaving a legacy in the crown shapes. This is why the crown that is
manually outlined around the big tree tends to be farther away from the crown space that is computed by this algorithm.

In conclusion, circle may not be a good representation for big tree. However, the current algorithm may be used to help an expert with choosing the radius for trees for better result.

**Baskett Sycamore stand**

Figure 4.6: The original image of American sycamore (platanus occidentalis L.) standat the Basket Wildlife Research Area near Ashland Missouri.
Figure 4.7: The original image with crown polygons and centers marked by an expert.

Figure 4.8: The weighted constrained Voronoi diagram calculated from center and radii with tree crown drawn in red.
Baskett, Bob Veech Road stand

Figure 4.9: The original image of Bob Veech Road stand at the Baskett Wildlife Research Area near Ashland Missouri.

Figure 4.10: The original image with crown polygons and centers marked by an expert.
Figure 4.11: The weighted constrained Voronoi diagram calculated from centers and radii with tree crown drawn in red.

**Baskett Walnut Road stand**

Figure 4.12: The original image of Walnut Road stand, at the Basket Wildlife Research Area near Ashland Missouri.
Figure 4.13: The original image with crown polygons and centers marked by an expert.

Figure 4.14: The weighted constrained Voronoi diagram calculated from centers and radii with tree crown drawn in red.
4.3 Biomedical application of Voronoi diagrams

Two biology image datasets are used for testing the weighted constrained Voronoi diagram. The first one is histopathology data and the second is wound image data. In histopathology data, Figure 4.15 the dark purple is nuclei. The center of each nucleus is used as a center of the circle and the radius for each circle is extracted by the size of each nucleus. As for wound image, Figure 4.15, the center of each mask is used as a generator location and the distance from center to the furthest away point within the same mask is used as a weight of that mask.

Results support that the weighted constrained Voronoi diagram can also be used with other fields.

Figure 4.15: Histopathology data provided by Michael Feldman (Department of Surgical Pathology, University of Pennsylvania) [9].
Figure 4.16: Extracted data from histopathology data set. Black dot is a center of the cell and the red circle represent its boundary.

Figure 4.17: The weighted constrained Voronoi diagram on histopathology data.
Fig 4.18: A mask of wound image with four unique foreground colors obtained from Nath et al. algorithm [10].

Figure 4.19: An extracted data image from wound image.
4.4 Complexity analysis

The current run time complexity of this algorithm can definitely be improved. For example, the method to search for overlapping generator is currently an exhaustive search which is considered an expensive operation. However, the current run time complexity of this algorithm is sufficient to handle forestry datasets. Figure 4.21 illustrates a dataset containing 10,000 random generators on image of dimension, 1000x1000, with the generators having weights randomly generated in the range [0, 50]. Executive on this dataset, which is considered unrealistically large by forestry standards, is completed in under a minute.
Table 4.1: The overall computation time and number of hidden generators for different sizes of datasets on the same area of space

<table>
<thead>
<tr>
<th>Number of generators</th>
<th>Overall computation time (ms)</th>
<th>Number of surviving generator</th>
<th>Estimated density (number of generator/size of space)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>14</td>
<td>167</td>
<td>0.0002</td>
</tr>
<tr>
<td>500</td>
<td>88</td>
<td>373</td>
<td>0.0005</td>
</tr>
<tr>
<td>1000</td>
<td>334</td>
<td>590</td>
<td>0.0010</td>
</tr>
<tr>
<td>1500</td>
<td>774</td>
<td>686</td>
<td>0.0015</td>
</tr>
<tr>
<td>2000</td>
<td>1528</td>
<td>830</td>
<td>0.0020</td>
</tr>
<tr>
<td>5000</td>
<td>11443</td>
<td>1188</td>
<td>0.0050</td>
</tr>
<tr>
<td>10000</td>
<td>63094</td>
<td>1570</td>
<td>0.0100</td>
</tr>
</tbody>
</table>

From Table 4.2, the time to search for overlapping is higher than the time takes to estimate new boundary when the dataset is not dense. When the dataset is denser or having more generators in the space, time takes to estimate new boundary start taking longer than time to search for overlapping generator. This is because time takes to estimate new boundary including time takes to compute oriented chord and interior intersecting point, validate intersecting point and boundary point and construct the final polygon depends on the number of overlapping generator which depends on the density of the dataset.
Table 4.2: The table for time to search for overlapping generator, time to estimate new boundary, and estimated density on different sizes of datasets on the same area of space

<table>
<thead>
<tr>
<th>Number of generators</th>
<th>Time to search for overlapping generator (ms)</th>
<th>Time to estimate new boundary (ms)</th>
<th>Estimated density (number of generator/ size of space)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>9</td>
<td>5</td>
<td>0.0002</td>
</tr>
<tr>
<td>500</td>
<td>58</td>
<td>30</td>
<td>0.0005</td>
</tr>
<tr>
<td>1000</td>
<td>236</td>
<td>98</td>
<td>0.0010</td>
</tr>
<tr>
<td>1500</td>
<td>525</td>
<td>249</td>
<td>0.0015</td>
</tr>
<tr>
<td>2000</td>
<td>987</td>
<td>541</td>
<td>0.0020</td>
</tr>
<tr>
<td>5000</td>
<td>5812</td>
<td>5631</td>
<td>0.0050</td>
</tr>
<tr>
<td>10000</td>
<td>23744</td>
<td>39350</td>
<td>0.0100</td>
</tr>
</tbody>
</table>

In conclusion, the obtained results are encouraging. The algorithm allocates space as intended. We have compared the different properties of the weighted constrained Voronoi diagram to other space allocations and those properties are letting a generator disappear or vanish when all of its area is occupied by other generators and are removing unnecessary large space around the edges. Also, we have shown that the weighted constrained Voronoi diagram can be applied to other fields. The weighted constrained Voronoi diagram can certainly be more beneficial in other fields with further study.
CHAPTER 5
SYLVAN-APA SOFTWARE TOOL FOR FORESTRY APPLICATIONS

Sylvan-APA is a C++ and Qt GUI-based software tool for interactively computing weighted Voronoi diagrams. The development of the Sylvan-APA tool was motivated by the need to calculate the Area Potentially Available parameter in landscape and tree stand simulation studies.

Sylvan-APA was built in order for foresters to use to compute and visualize Area Potentially Available for the stand that foresters have measured or created through Sylvan-APA interface. Again, the weighted constrained Voronoi diagram is designed to be as representative as possible to Area Potentially Available for trees. Foresters normally do not have a strong background in using computers. Hence, Sylvan-APA user interface has to be presented in an organized manner and well designed so that it will intuitively guide users how to use Sylvan-APA thoroughly.

To understand Sylvan-APA, Sylvan-APA is divided into three main parts which are input, display and output.

5.1 Sylvan-APA input

Sylvan-APA can take two types of input. First, users can interactively add trees to the space by clicking on Sylvan-APA display. The first click is the center of the tree. The
distance from the first click to the second click is computed and then used as a weight or radius of the tree. Second, users can load comma separated value file. Each row in the file should contain x coordinate, y coordinate and weight as a radius consequently (see Section A.F).

5.2 Sylvan-APA display

Sylvan-APA display is mainly used to demonstrate results of the weighted constrained Voronoi diagram. Figure 5.1 illustrates weighted constrained Voronoi

Figure 5.1: An example of weighted constrained Voronoi diagram being drawn in Sylvan-APA.
diagram on one of the dataset. Also, each polygon or circle on Sylvan-APA display is a selectable object and users can perform certain tasks on selected polygons. For example, users can delete or configure part of polygon to be displayed or hidden. APA display on Figure 5.2 illustrates what options are to be displayed for each polygon.

Additionally, there are few other options for user to configure on Sylvan-APA display. Users can specify whether to draw using OpenGL, to use an aliasing drawing, or to adjust the size of the plot.

![Preference Dialog](image)

Figure 5.2: Sylvan-APA preference dialog.
5.3 Sylvan-APA output

There are three types of output that users can use for different propose: in Sylvan-APA: screen printing, tree data saving, and APA data saving.

- Screen printing is used to save whatever is on display as an image file.
- Tree data saving – The file format that is used to save tree data is the same as the input file format. This type of output is used when the user creates their own tree stands or after the user modifies their tree stands.
- Tree crown data saving – This option is for users to save the crown estimated by weighted constrained Voronoi diagram. Each tree crown polygon is divided into multiple points based on its azimuth to the center of the circle because this then can be easily used to determine how fast tree will grow at each azimuth. There are two types of files that can output tree crown data saving. First is R file (*.r). This type of file can be opened and drawn in R, the statistical computing tool with graphics, without any modification on the file. Another one is called azimuth file (*.azi). This type of file has all the information related to the crown tree polygon including centroid and area.
CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In conclusion, the weighted constrained Voronoi diagram allocates the space correctly as expected. It solves the problem in which the area of a point that is close to the edge of the domain can be unnecessarily large. A logical assumption is that the objects can only use some of the space and the weight is used to describe that limit. Also, the weighted constrained Voronoi diagram allows objects to disappear if all of the area it has is occupied by other generators. This constraint makes perfect senses in some scenarios. For an example, tree can die, if they have no growing space left.

In computational geometry, special cases are always problematic. The weighted constrained Voronoi diagram however can handle all kinds of intersection as expected. The weighted constrained Voronoi diagram can be used to allocate the space of any non-mixing object that can be modeled as a circle. For example, an area potentially available for a tree can be modeled as a circle and the area of that tree cannot be mixed. Hence, the weighted constrained Voronoi diagram works perfectly for it. Another example would be biological cells. Again, cells can be modeled as circles and they are non-mixing objects. Hence, the weighted constrained Voronoi diagram can also be used to allocate the space for a biological cell. What we have achieved is the weighted constrained Voronoi
The weighted constrained Voronoi diagram can be used in real application as stated above. The weighted constrained Voronoi may also work in other fields if further study is made.

6.2 Future work

For every piece of work and every algorithm, there always is something that can be added or modified to make it more efficient, more generalized, or more functionality.

The first improvement that can be made is to investigate if the weighted constrained Voronoi diagram can be applied to any other geometric shapes. As for current forestry technology, the circle is the best representation of a tree growth model. An ellipse is the best representation for a cell. Other geometric shapes such as a square, a rectangle, an octagon, a hexagon, a heptagon, may be good for representing other kind of object.

Second, finding the intersecting generators is currently exhaustive search. This can be improved by sorting the generators by size of radii and then processing the circles in decreasing size order. This can potentially speed up the algorithm because if the generator is hidden or its space is completely occupied by other generators, there is no further calculation needed for it and bigger generators occupy more space and potentially have more generators hidden in it. Also, line-sweep algorithm can be applied in order to fix the use of exhaustive search.

Third, the current method of region selection in Sylvan-APA is rectangle based method because it is built in function provided by Qt and the rectangle is a boundary for
each generator before the space is allocated. When weighted constrained Voronoi is
displayed, region selection method will be working better if we fit the rectangle to new
boundary.
REFERENCE


APPENDIX A

USER’S MANUAL
A.1 Introduction

The Sylvan-APA application allows users to load a file or interactively add generators to the space. The files that can be loaded are in comma-separated value file format. After a set of generators has been loaded or added to the space, users can then visualize weighted constrained Voronoi diagram. Also, this application provides other several views of the data set.

A.2 Sylvan-APA main window

Sylvan-APA main window consists of four parts which are main view, menu bar, tool bar and status bar. Main view provides the visualization for weighted constrained Voronoi diagram and other several views of the dataset. Menu provides all the access to all of the functionalities there are in Sylvan-APA including the help file. On the other hands, the tool bar provides an access to the functionalities that are often used. Status bar displays the position of user’s mouse.
A.3 Sylvan-APA preference dialog

Sylvan-APA preference dialog allows users to configure the various visualization on the main view from the whole view level down to each generator level. Each generator consists of several elements and those elements are centroid, center, a line from center to centroid, ID, new boundary, original boundary and oriented chords. From Figure A.2, APA display group box allows users to toggle whether to display any of the elements in the generator. Configuration group box allows users to configure the top level of the
visualization. Through configuration group box, users can configure whether to use openGL, antialiased drawing, whether to hose hidden tree and main view size. Lastly, set tree color button and set hidden tree color button are used to set the color for generators and hidden generators accordingly.

![Sylvan-APA preference dialog](image)

Figure A.2: Sylvan-APA preference dialog.
A.4 Menus, Toolbars and Status bar

A.4.1 Menus

File: This menu provides option for loading dataset, saving dataset, and screen captures the main view.

Load: Load a dataset.

Save: Save a dataset.

Export: Screen capture what is being displayed on the main view.

Exit: Close the application.

Edit: This menu provides option for editing the visualization on the main view.

Draw APA: Draw weighted constrained Voronoi diagram on the main view.

Draw Circle: Draw the generator and its weight on the main view.

Clear: Clear all generators on the space

Mode: This menu provides options for users to toggle between ADD and SELECT mode

Add: Use adding mode. This mode allows users to interactively add generators to the main view.

Select: Use selecting mode. This mode allows users to select a particular generator by clicking on it and further configure its visualization.

Option: This menu provides an ability to open a preference dialog
**Preference**: Pop up the preference dialog

**Help**: This menu provides an access to help dialog

**About Sylvan-APA**: Pops up help dialog

### A.4.2 Toolbars

**Load**: Pop a dialog that allows users to select to load a dataset file.

**Save**: Pop a dialog that allows users to save the generator on the space as a comma-separated value file.

**Draw APA**: Draw weighted constrained Voronoi diagram on the main view.

**Draw Circle**: Draw the generator and its weight on the main view.

**Center scene**: Make sure that all items on the scene are visible and fit it in the view.

**Clear**: Remove all the generators on the space.

### A.4.3 Status bar

Status bar is located at the bottom of the main window and is used to display the location in coordinate of the mouse of the user.

### A.5 Keyboard Shortcuts

There are few shortcuts that mainly used to provide simplify frequently used operations such as displaying weighted constrained Voronoi diagram, displaying the generator and its weight, or clear the dataset.
W: Zooming in.

S: Zooming out.

A: Visualizing weighted constrained Voronoi diagram on the set of generator on the space.

G: Visualizing the generator and its weight on the space.

C: Clearing all generators on the space.

X: Drawing all generators and its weight with all of its oriented chords.

A.6 File formats

Sylvan-APA can load handle three different comma-separated value file formats. We have the different extension for each file format even though they are all comma-separated value file. The file extensions are csv, R and azi.

CSV: This type of file is used for loading and saving the generator and its weight. Each row consists of center of the generator x, center of the generator y and the weight of the generator.

64.12, 688.39, 28.98
909.94, 350.04, 25.46
728.18, 763.95, 8.27
946.11, 885.71, 9.3
555.27, 249.91, 45.6
596.74, 9.4, 30.24
392.27, 553.04, 35.55
165.19, 820.47, 47.17
109.53, 728.97, 38.95
761.96, 153.78, 30.97
23.65, 80.23, 36.1
55.76, 242.71, 2.31
R: This one is used for outputting the weighted constrained Voronoi diagram result into the format that is readily to be read by R (language and environment for statistical computing and graphics). Data for each generator is separated by NA, NA and data for each generator consists of the position of new boundary sampled azimuth. In here, we sample every 10 azimuth. The following is an example of R file.

\[
\begin{array}{l}
x, y \\
188.84659, 196 \\
187.60652, 204.39431 \\
186.3542, 212.87154 \\
185.00786, 221.98527 \\
183.45945, 232.46677 \\
178.56729, 241.96264 \\
170.00005, 247.9615 \\
160.52127, 252.38154 \\
150.41896, 255.08845 \\
140.00008, 256 \\
129.5812, 255.08848 \\
119.47888, 252.38159 \\
110.00009, 247.96158 \\
101.43283, 241.96274 \\
94.037413, 234.56735 \\
88.038542, 226.00011 \\
83.618491, 216.52134 \\
80.911561, 206.41904 \\
80, 196.00016 \\
80.911506, 185.58127 \\
83.618382, 175.47896 \\
88.038383, 166.00016 \\
94.037208, 157.43289 \\
101.43259, 150.03746 \\
109.99982, 144.03858 \\
119.47858, 139.61852 \\
130.24396, 140.67197 \\
139.99981, 147.05543 \\
147.73735, 152.11826 \\
154.38766, 156.4697 \\
160.50983, 160.47557 \\
166.51255, 164.40328 \\
172.77321, 168.49975
\end{array}
\]
AZI: This one is used currently for human to read information for each generator and the information includes center, weight, centroid, area and coordinate or each azimuth. The following is the example of AZI file. Similarly to R file format, AZI contains the position
of new boundary at the sampled azimuth. Again, we also sample every 10 azimuth in
here.

#
id, cx, cy, radius
0, 140, 196, 60
#
id, centroidx, centroidy, area
0, 134.081, 200.323, 9587.32
#
azimuth, x, y
0, 188.847, 196
10, 187.607, 204.394
20, 186.354, 212.872
30, 185.008, 221.985
40, 183.459, 232.467
50, 178.567, 241.963
60, 170, 247.961
70, 160.521, 252.382
80, 150.419, 255.088
90, 140, 255.088
100, 129.581, 255.088
110, 119.479, 252.382
120, 110, 247.962
130, 101.433, 241.963
140, 94.0374, 234.567
150, 88.0385, 226
160, 83.6185, 216.521
170, 80.9116, 206.419
180, 80, 196
190, 80.9115, 185.581
200, 83.6184, 175.479
210, 88.0384, 166
220, 94.0372, 157.433
230, 101.433, 150.037
240, 110, 144.039
250, 119.479, 139.619
260, 130.244, 140.672
270, 140, 147.055
280, 147.737, 152.118
290, 154.388, 156.47
300, 160.51, 160.476
310, 166.513, 164.403
320, 172.773, 168.5
330, 179.738, 173.057
340, 188.065, 178.505
350, 190.153, 187.156
360, 188.847, 196
#
id, cx, cy, radius
1, 193, 115, 71
<table>
<thead>
<tr>
<th>id, centroidx, centroidy, area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 185.711, 107.371, 12361.3</td>
</tr>
</tbody>
</table>

# azimuth, x, y
0, 237.31, 115
10, 235.255, 122.451
20, 233.267, 129.656
30, 231.223, 137.068
40, 228.982, 145.192
50, 226.347, 154.742
60, 222.984, 166.933
70, 213.777, 172.083
80, 203.725, 175.826
90, 193.179.819
100, 182.45, 174.832
110, 173.383, 168.899
120, 165.035, 163.437
130, 156.851, 158.081
140, 148.314, 152.496
150, 138.818, 146.282
160, 127.464, 138.853
170, 123.079, 127.329
180, 122.115
190, 123.079, 102.671
200, 126.282, 90.7168
210, 131.512, 79.5002
220, 138.611, 69.3623
230, 147.362, 60.611
240, 157.5, 53.5123
250, 168.716, 48.2819
260, 180.671, 45.0787
270, 193.44
280, 205.329, 45.0786
290, 217.283, 48.2817
300, 228.5, 53.512
310, 238.638, 60.6106
320, 247.389, 69.3618
330, 245.705, 84.5706
340, 242.256, 97.072
350, 239.576, 106.787
360, 237.31, 115