Generalized Voronoi Diagrams: Theory and Related Applications

Stefan de Lorenzo
University of Salzburg, Department of Computer Science

October 5, 2021

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Overview

An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

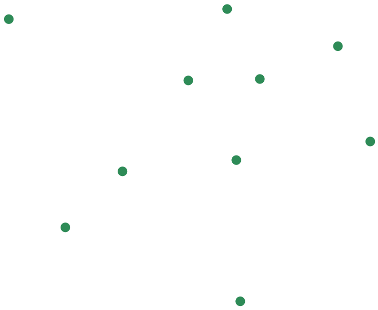
Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

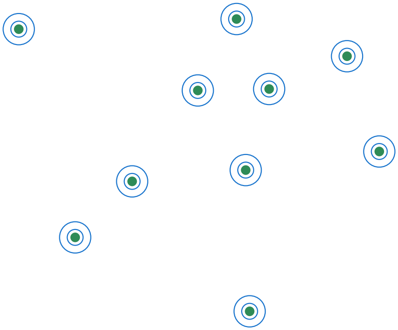
On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*

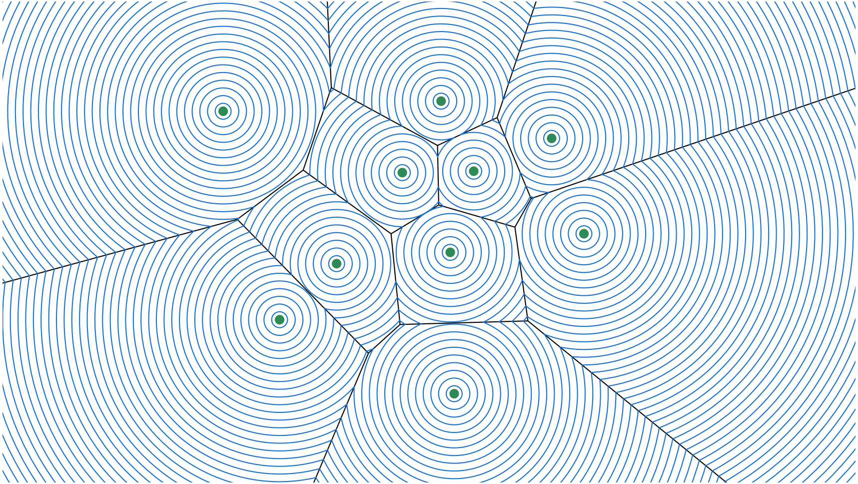
Wavefront Propagation





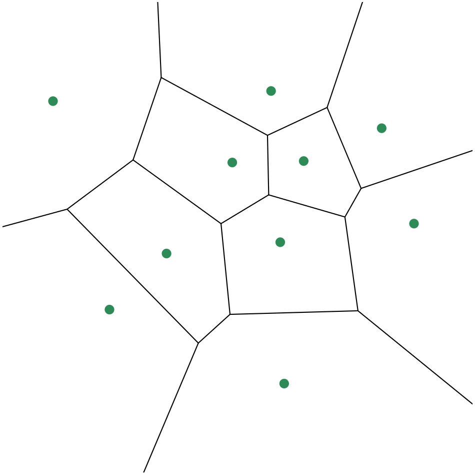
The ***wavefront*** $\mathcal{WF}(t)$ at time t consists of ***wavefront arcs*** and ***wavefront vertices***.

Wavefront Propagation



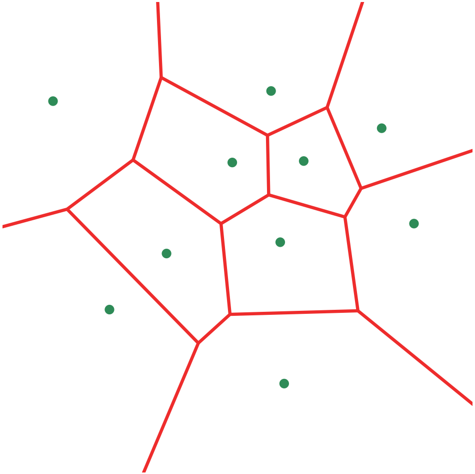
The ***wavefront*** $\mathcal{WF}(t)$ at time t consists of ***wavefront arcs*** and ***wavefront vertices***.

The Voronoi Diagram of Points

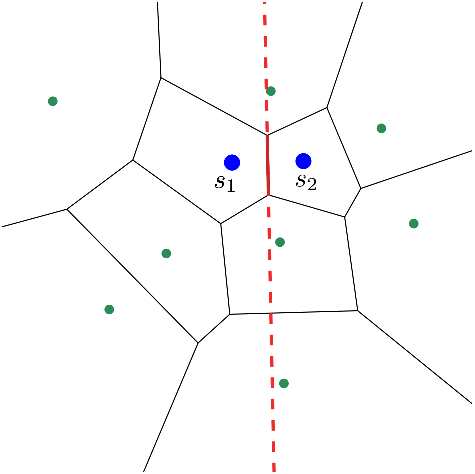


- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.

The Voronoi Diagram of Points

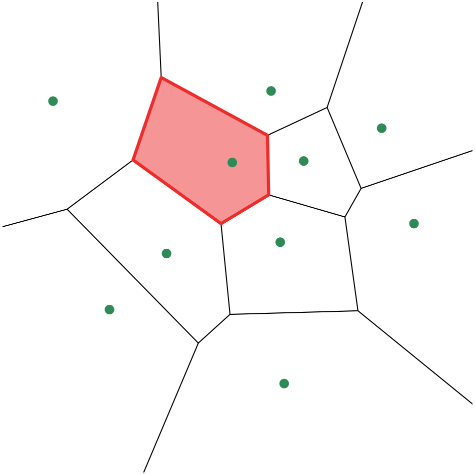


- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.
- It consists of ***Voronoi edge*** and ***Voronoi nodes***.



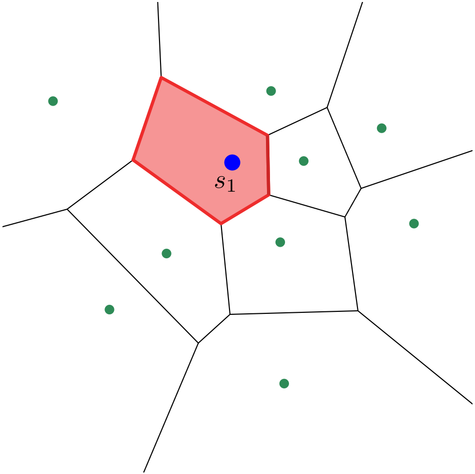
- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.
- It consists of ***Voronoi edge*** and ***Voronoi nodes***.
- Each Voronoi edges is situated on a ***bisector*** that is defined by a pair of input sites.

The Voronoi Diagram of Points

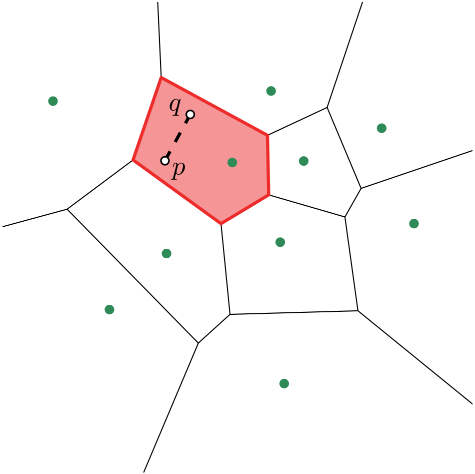


- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.
- It consists of ***Voronoi edge*** and ***Voronoi nodes***.
- Each Voronoi edges is situated on a ***bisector*** that is defined by a pair of input sites.
- The Voronoi diagram subdivides the Euclidean space into ***Voronoi regions***.

The Voronoi Diagram of Points

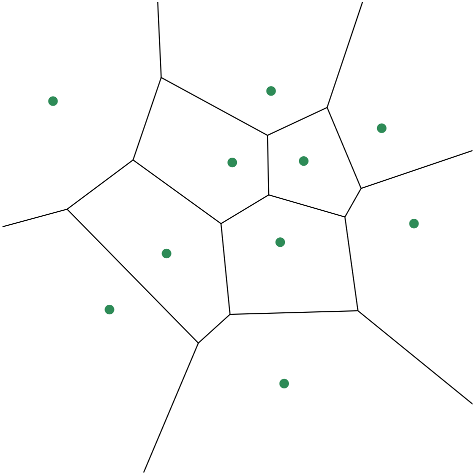


- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.
- It consists of ***Voronoi edge*** and ***Voronoi nodes***.
- Each Voronoi edges is situated on a ***bisector*** that is defined by a pair of input sites.
- The Voronoi diagram subdivides the Euclidean space into ***Voronoi regions***.
- Every input site is associated with such a region.



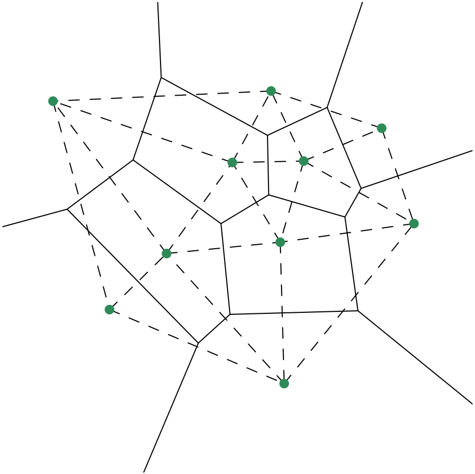
- Let S be a set of ***input sites*** in the plane.
- The ***Voronoi diagram*** $\mathcal{VD}(S)$ is a versatile tool in computational geometry.
- It consists of ***Voronoi edge*** and ***Voronoi nodes***.
- Each Voronoi edges is situated on a ***bisector*** that is defined by a pair of input sites.
- The Voronoi diagram subdivides the Euclidean space into ***Voronoi regions***.
- Every input site is associated with such a region.
- Each Voronoi region is ***convex***.

The Delaunay Triangulation

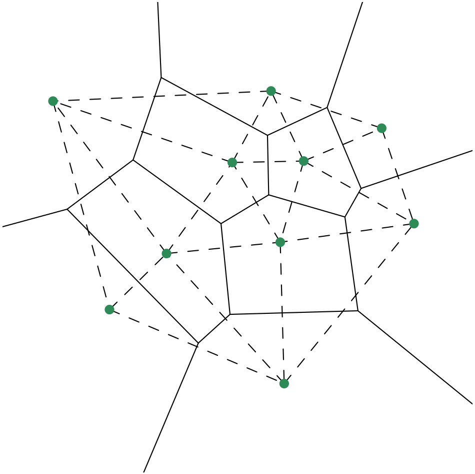


- The ***Delaunay triangulation*** can be derived in linear time from the corresponding Voronoi diagram.

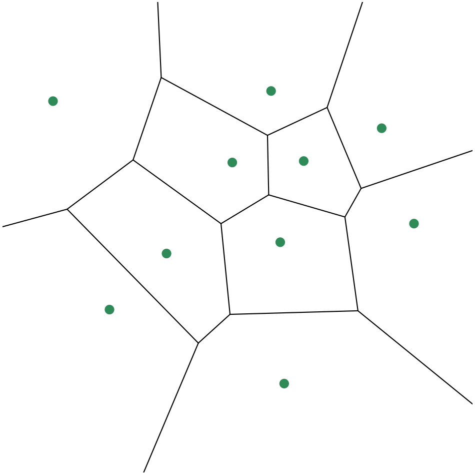
The Delaunay Triangulation



- The ***Delaunay triangulation*** can be derived in linear time from the corresponding Voronoi diagram.
- Neighbors in $\mathcal{VD}(S)$ are connected via a triangulation edge.
- The Delaunay triangulation is the dual graph of the Voronoi diagram.



- The ***Delaunay triangulation*** can be derived in linear time from the corresponding Voronoi diagram.
- Neighbors in $\mathcal{VD}(S)$ are connected via a triangulation edge.
- The Delaunay triangulation is the dual graph of the Voronoi diagram.
- It maximizes the minimum angle inside a triangle over all possible triangulations.



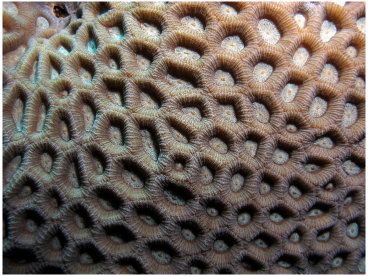
Theorem (Shamos and Hoey 1975)

The Voronoi diagram of n input points can be computed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space using a divide-and-conquer strategy.

Theorem (Fortune 1987)

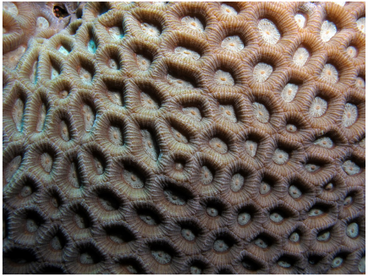
The Voronoi diagram of n input points can be computed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space using a sweepline algorithm.

Voronoi Diagrams in Nature



Credit: *Keats 2009*

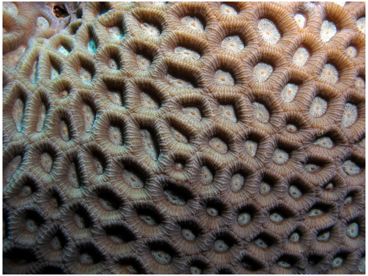
Voronoi Diagrams in Nature



Credit: *Keats 2009*

Credit: *Hillewaert 2010*

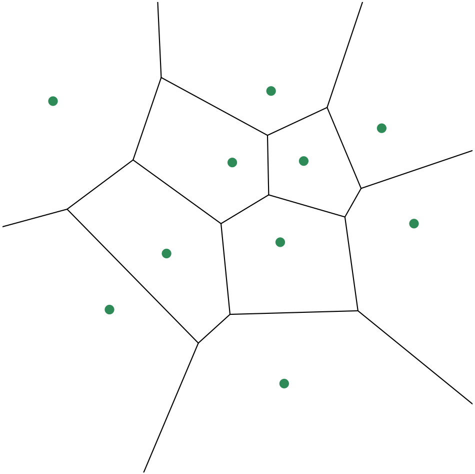
Voronoi Diagrams in Nature



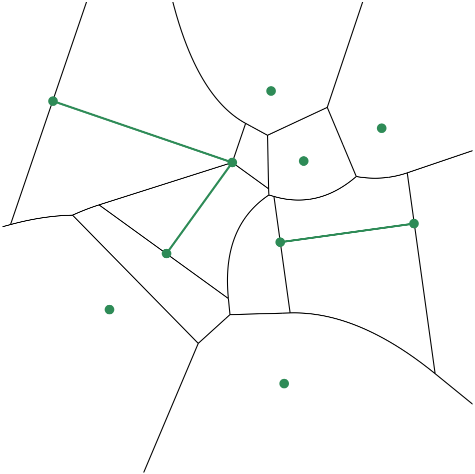
Credit: *Keats 2009*

Credit: *Hillewaert 2010*

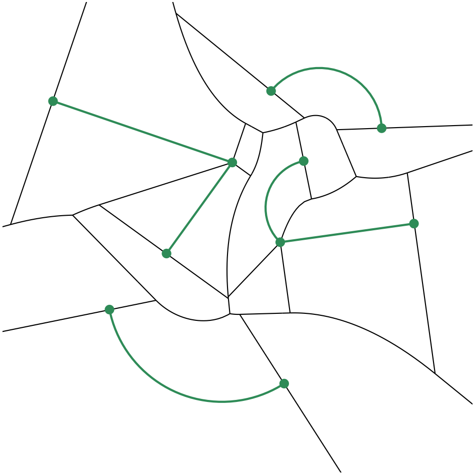
Credit: *Rader 2009*



The Voronoi diagram can be generalized by allowing other types of input sites such as ...

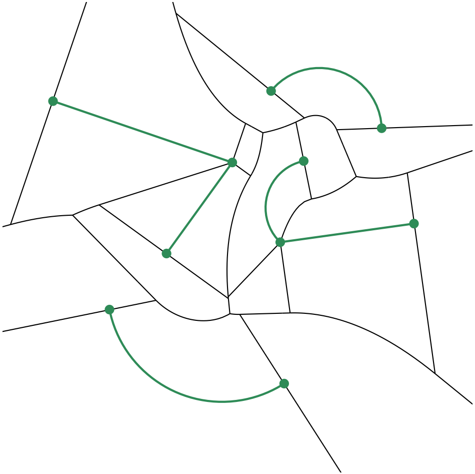


The Voronoi diagram can be generalized by allowing other types of input sites such as straight-line segments ...



The Voronoi diagram can be generalized by allowing other types of input sites such as straight-line segments and circular arcs.

Allowing Other Types of Input Sites



The Voronoi diagram can be generalized by allowing other types of input sites such as straight-line segments and circular arcs.

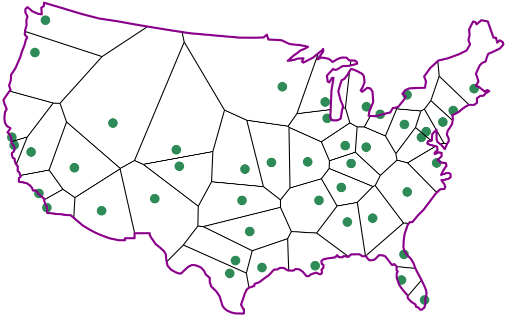
Theorem (Yap 1987)

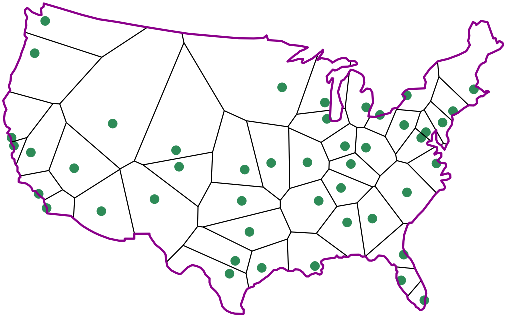
It is possible to generate the Voronoi diagram of n straight-line segments and circular arcs in optimal $\mathcal{O}(n \log n)$ time.

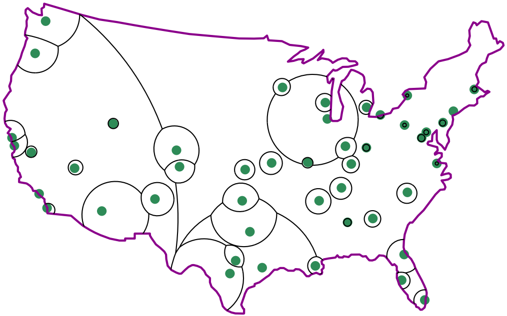
Theorem (Held and Huber 2009)

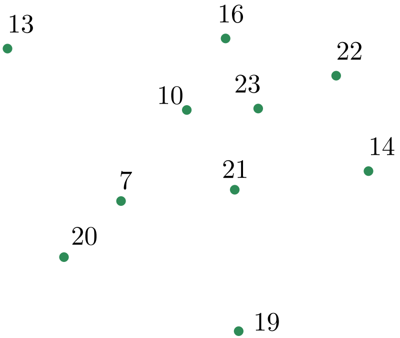
The Voronoi diagram of n points, straight-line segments, and circular arcs can be computed in expected $\mathcal{O}(n \log n)$ time.

Multiplicatively Weighted Voronoi Diagrams

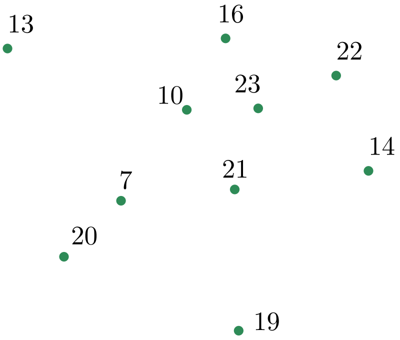








- Consider a set S of point sites in the Euclidean plane.
- We associate every site s with a real-valued ***weight*** $w(s) > 0$.

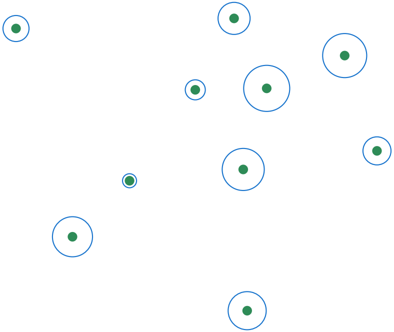


- Consider a set S of point sites in the Euclidean plane.
- We associate every site s with a real-valued ***weight*** $w(s) > 0$.

Definition (Weighted Distance)

The ***(multiplicatively) weighted distance*** $d_w(p, s)$ between a weighted site $s \in S$ and a points $p \in \mathbb{R}^2$ is given by

$$d_w(p, s) := \frac{d(p, s)}{w(s)}.$$

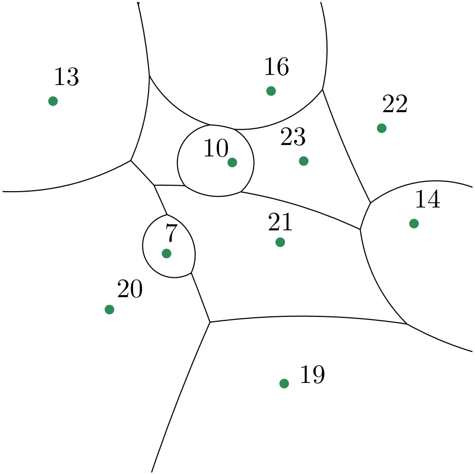


- Consider a set S of point sites in the Euclidean plane.
- We associate every site s with a real-valued ***weight*** $w(s) > 0$.
- The ***offset circles*** expand at different rates.

Definition (Weighted Distance)

The ***(multiplicatively) weighted distance*** $d_w(p, s)$ between a weighted site $s \in S$ and a points $p \in \mathbb{R}^2$ is given by

$$d_w(p, s) := \frac{d(p, s)}{w(s)}.$$

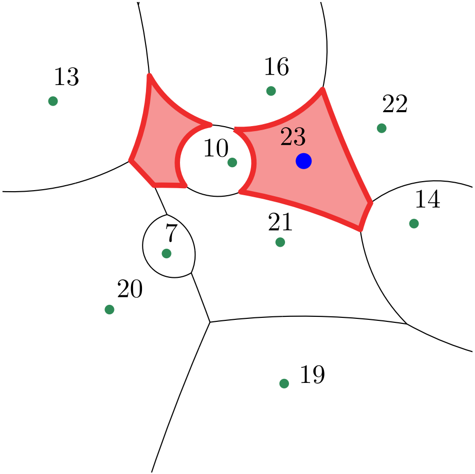


- Consider a set S of point sites in the Euclidean plane.
- We associate every site s with a real-valued ***weight*** $w(s) > 0$.
- The ***offset circles*** expand at different rates.
- A Voronoi edge is (in general) formed by circular arcs.

Definition (Weighted Distance)

The ***(multiplicatively) weighted distance*** $d_w(p, s)$ between a weighted site $s \in S$ and a points $p \in \mathbb{R}^2$ is given by

$$d_w(p, s) := \frac{d(p, s)}{w(s)}.$$

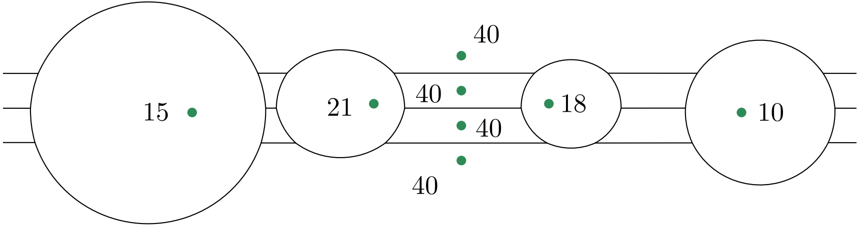


- Consider a set S of point sites in the Euclidean plane.
- We associate every site s with a real-valued ***weight*** $w(s) > 0$.
- The ***offset circles*** expand at different rates.
- A Voronoi edge is (in general) formed by circular arcs.
- The Voronoi regions are (possibly) disconnected.

Definition (Weighted Distance)

The ***(multiplicatively) weighted distance*** $d_w(p, s)$ between a weighted site $s \in S$ and a points $p \in \mathbb{R}^2$ is given by

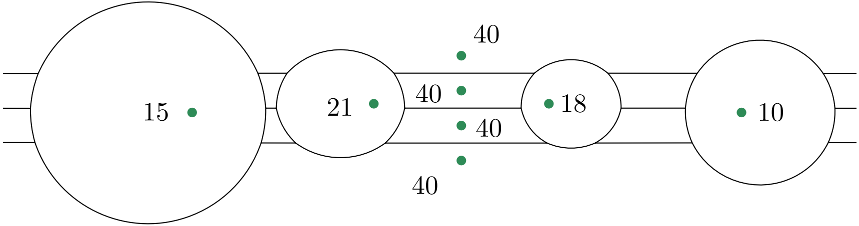
$$d_w(p, s) := \frac{d(p, s)}{w(s)}.$$



Theorem (Aurenhammer and Edelsbrunner 1984)

The multiplicatively weighted Voronoi diagram of n input sites has a combinatorial complexity of $\mathcal{O}(n^2)$ in the worst case.

A Worst-Case Example



Theorem (Aurenhammer and Edelsbrunner 1984)

The multiplicatively weighted Voronoi diagram of n input sites has a combinatorial complexity of $\mathcal{O}(n^2)$ in the worst case.

Theorem (Aurenhammer and Edelsbrunner 1984)

The multiplicatively weighted Voronoi diagram of n input sites can be computed in (optimal) $\mathcal{O}(n^2)$ time and space.

Overview

An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

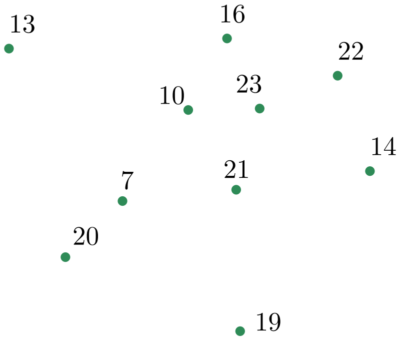
Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

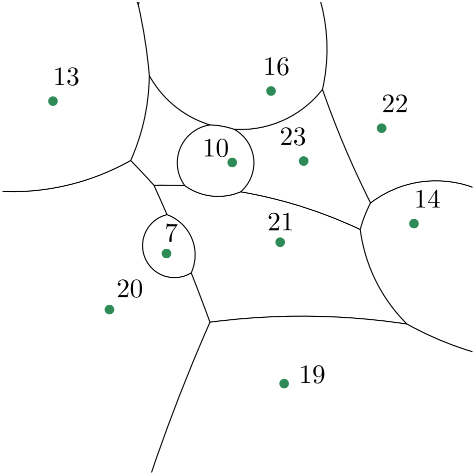
Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*

A Wavefront-Based Strategy



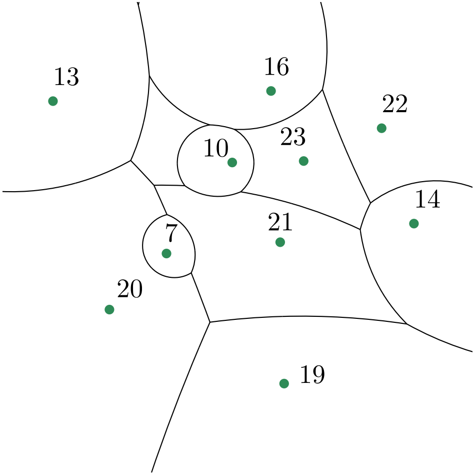
Problem

Given: A set S of n input points in the plane, where every $s \in S$ is associated with a weight $w(s) > 0$.



Problem

Given: A set S of n input points in the plane, where every $s \in S$ is associated with a weight $w(s) > 0$.
Find: The multiplicatively weighted Voronoi diagram (MWVD) $\mathcal{VD}_w(S)$ of S.

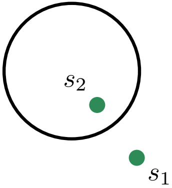


Problem

Given: A set S of n input points in the plane, where every $s \in S$ is associated with a weight $w(s) > 0$.
Find: The multiplicatively weighted Voronoi diagram (MWVD) $\mathcal{VD}_w(S)$ of S.

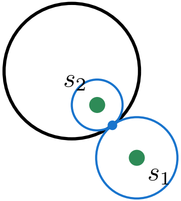
- We present a wavefront-based approach for computing the MWVD.
- The wavefront covers an increasing portion of the plane over time.
- It consists of wavefront arcs and wavefront vertices.
- Whenever a wavefront arc appears or disappears, a new Voronoi node is discovered.

Offset Circles



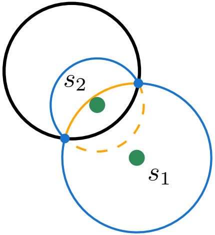
- Every site is associated with an ***offset circle***.

Offset Circles



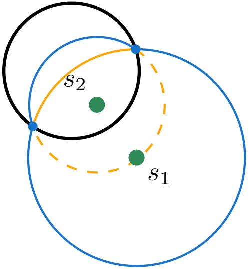
- Every site is associated with an ***offset circle***.
- Two ***moving vertices*** trace out the bisector as time progresses.

Offset Circles



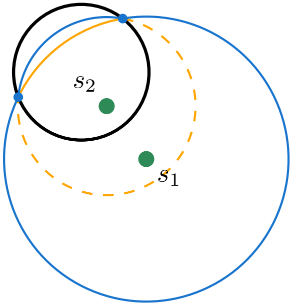
- Every site is associated with an ***offset circle***.
- Two ***moving vertices*** trace out the bisector as time progresses.
- ***Inactive arcs*** along the offset circles are eliminated.
- The ***active arcs*** are stored in sorted angular order.

Offset Circles



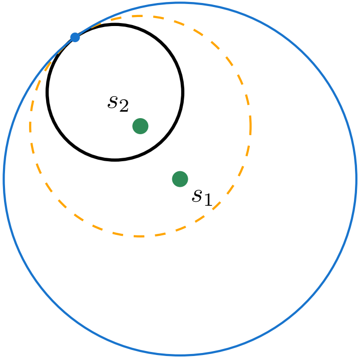
- Every site is associated with an ***offset circle***.
- Two ***moving vertices*** trace out the bisector as time progresses.
- ***Inactive arcs*** along the offset circles are eliminated.
- The ***active arcs*** are stored in sorted angular order.

Offset Circles

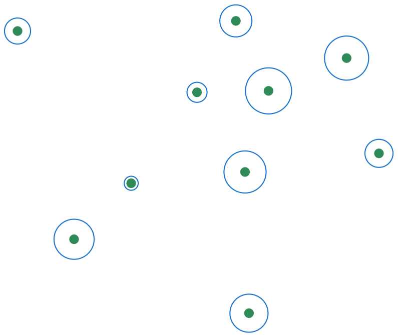


- Every site is associated with an ***offset circle***.
- Two ***moving vertices*** trace out the bisector as time progresses.
- ***Inactive arcs*** along the offset circles are eliminated.
- The ***active arcs*** are stored in sorted angular order.

Offset Circles



- Every site is associated with an ***offset circle***.
- Two ***moving vertices*** trace out the bisector as time progresses.
- ***Inactive arcs*** along the offset circles are eliminated.
- The ***active arcs*** are stored in sorted angular order.



- ***Collision*** and ***domination events*** mark the initial and last contact of two offset circles.
- ***Arc events*** happen whenever active arcs appear or disappear.
- These events are stored in a priority queue.
- The angular order of active arcs only changes at events.

Event Handling

- All topological changes of the wavefront are properly detected.
- A quadratic number of collision events are computed in any case.
- A moving vertex can be charged with a constant number of arc events.
- In the worst case $\mathcal{O}(n^2)$ arc events take place.
- All events can be handled in $\mathcal{O}(\log n)$ time.

Event Handling

- All topological changes of the wavefront are properly detected.
- A quadratic number of collision events are computed in any case.
- A moving vertex can be charged with a constant number of arc events.
- In the worst case $\mathcal{O}(n^2)$ arc events take place.
- All events can be handled in $\mathcal{O}(\log n)$ time.

Theorem (Held and de Lorenzo 2020)

The MWVD $\mathcal{VD}_w(S)$ of a set S of n weighted point sites in $\mathbb{R}^2$ can be computed in $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n^2)$ space.

Event Handling

- All topological changes of the wavefront are properly detected.
- A quadratic number of collision events are computed in any case.
- A moving vertex can be charged with a constant number of arc events.
- In the worst case $\mathcal{O}(n^2)$ arc events take place.
- All events can be handled in $\mathcal{O}(\log n)$ time.

Theorem (Held and de Lorenzo 2020)

The MWVD $\mathcal{VD}_w(S)$ of a set S of n weighted point sites in $\mathbb{R}^2$ can be computed in $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n^2)$ space.

Theorem (Held and de Lorenzo 2020)

The MWVD $\mathcal{VD}_w(S)$ of a set S of n weighted point sites in one dimension can be computed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.

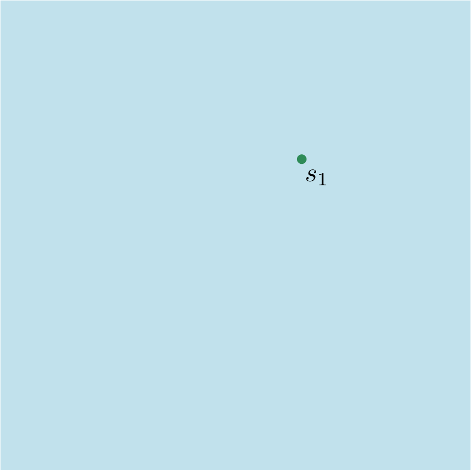
Reducing the Number of Collisions

Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.

Reducing the Number of Collisions

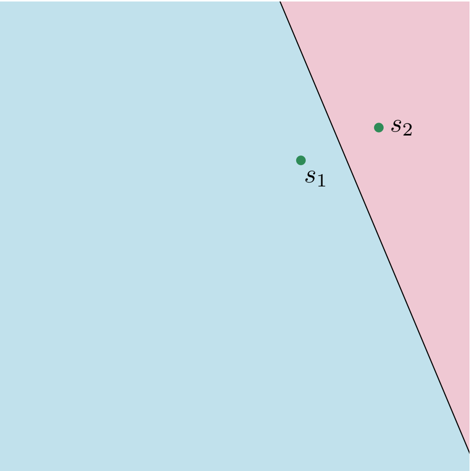


Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.

Reducing the Number of Collisions

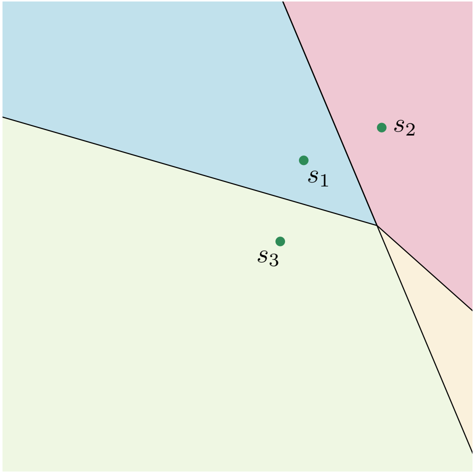


Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.

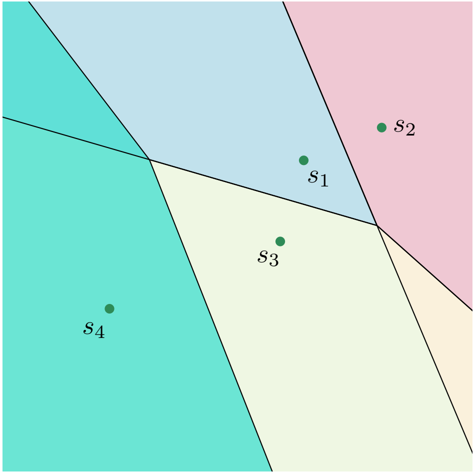
Reducing the Number of Collisions



Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.

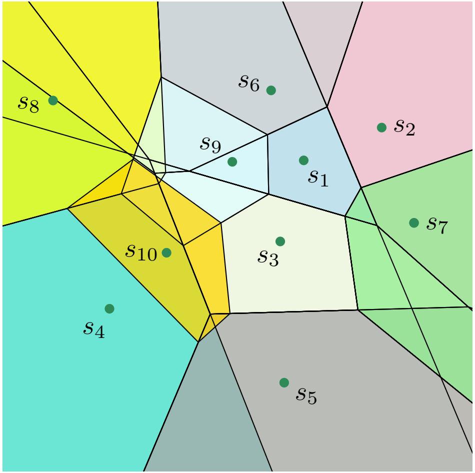


Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.

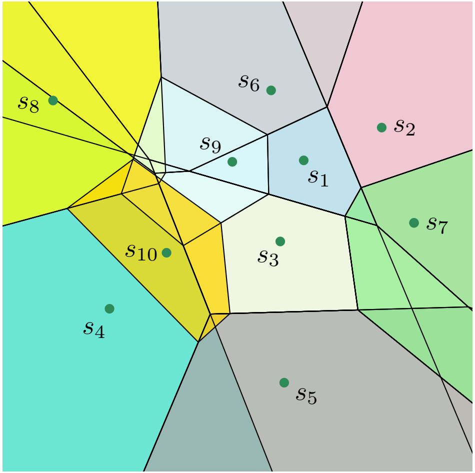
Reducing the Number of Collisions



Theorem (Har-Peled and Raichel 2015)

Let S be a set of n points in the plane, where for each point we independently sample a weight from some distribution. Then the expected complexity of the MWVD of S is $\mathcal{O}(n \log^2 n)$.

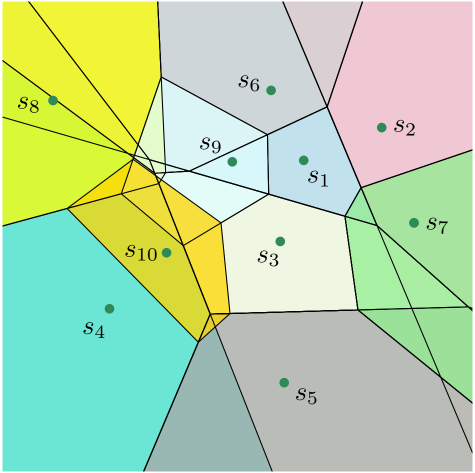
- A vast number of collisions are invalid for general input.
- The calculation of all possible collision requires a high computational effort.
- Invalid collision are filtered in an additional preprocessing step.
- The average case behavior of the algorithm is improved by using an ***overlay arrangement***.



Definition (Candidate Set)

The ***candidate set*** for a weighted nearest neighbor of $q \in \mathbb{R}^2$ consists of all sites $s \in S$ such that all other sites in S either have a smaller weight or a larger Euclidean distance to q.

Candidate Sets



Definition (Candidate Set)

The ***candidate set*** for a weighted nearest neighbor of $q \in \mathbb{R}^2$ consists of all sites $s \in S$ such that all other sites in S either have a smaller weight or a larger Euclidean distance to q.

- Only sites within the same candidate set may collide.

Lemma (Har-Peled and Raichel 2015)

For all points $q \in \mathbb{R}^2$, the candidate set for q among S is of size $\mathcal{O}(\log n)$ with high probability.

Overlay Arrangement

Lemma (Har-Peled and Raichel 2015)

For all points $q \in \mathbb{R}^2$, the candidate set for q among S is of size $\mathcal{O}(\log n)$ with high probability.

Theorem (Kaplan, Ramos, and Sharir 2011)

The expected complexity of overlay arrangement is bounded by $\mathcal{O}(n \log n)$.

Lemma (Har-Peled and Raichel 2015)

For all points $q \in \mathbb{R}^2$, the candidate set for q among S is of size $\mathcal{O}(\log n)$ with high probability.

Theorem (Kaplan, Ramos, and Sharir 2011)

The expected complexity of overlay arrangement is bounded by $\mathcal{O}(n \log n)$.

Theorem (Held and de Lorenzo 2020)

All collision events can be determined in $\mathcal{O}(n \log^3 n)$ expected time by computing the overlay arrangement of a set S of n input sites.

Lemma (Har-Peled and Raichel 2015)

For all points $q \in \mathbb{R}^2$, the candidate set for q among S is of size $\mathcal{O}(\log n)$ with high probability.

Theorem (Kaplan, Ramos, and Sharir 2011)

The expected complexity of overlay arrangement is bounded by $\mathcal{O}(n \log n)$.

Theorem (Held and de Lorenzo 2020)

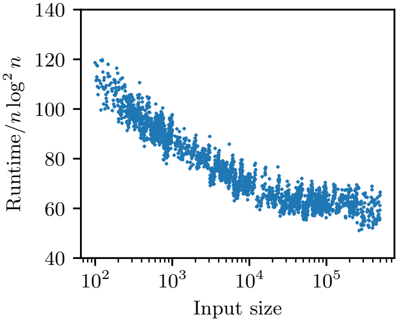
All collision events can be determined in $\mathcal{O}(n \log^3 n)$ expected time by computing the overlay arrangement of a set S of n input sites.

Theorem (Held and de Lorenzo 2020)

A wavefront-based approach allows to compute the MWVD of a set S of n (randomly) weighted point sites in expected $\mathcal{O}(n \log^4 n)$ time and expected $\mathcal{O}(n \log^3 n)$ space.

Experimental Evaluation

- The implementation is based on the Computational Geometry Algorithms Library (CGAL).
- We tested our strategy on over 3000 different inputs ranging from 256 vertices to 500 000 vertices.
- All tests were carried out on an Intel Core i9-7900X processor clocked at 3.3 GHz.
- For all of these inputs, the weights and point locations were chosen uniformly at random.



UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Overview

An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

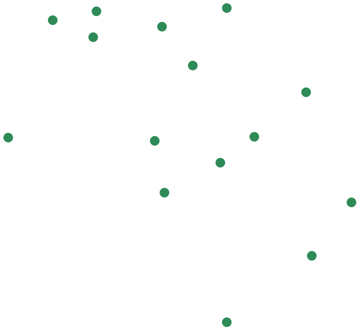
On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*

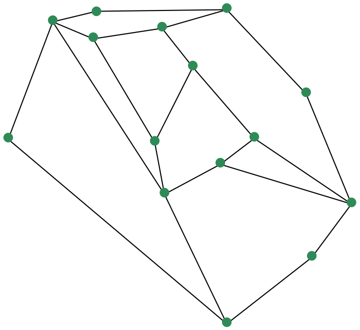
Problem

Given: A set S of n points in the plane.



Problem

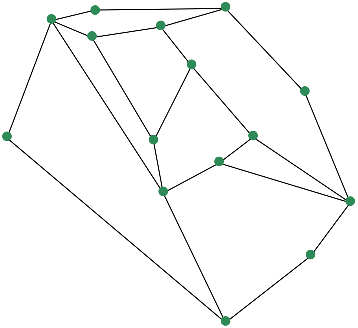
Given: A set S of n points in the plane.
Find: A plane graph with vertex set S (with each point in S having positive degree) that partitions the convex hull of S into the smallest possible number of convex faces. Note that collinear points are allowed on face boundaries, so all internal angles of a face are at most π.



Problem Specification

Problem

Given: A set S of n points in the plane.
Find: A plane graph with vertex set S (with each point in S having positive degree) that partitions the convex hull of S into the smallest possible number of convex faces. Note that collinear points are allowed on face boundaries, so all internal angles of a face are at most π.



CG:SHOP 2020

Download | Submit Solution

Organized by: Erik Demaine (MIT), Sándor Fekete (TU Braunschweig), Phillip Keldenich (TU Braunschweig), Dominik Krupke (TU Braunschweig), Joseph S. B. Mitchell (Stony Brook University)

25 teams participating
Sept. 30, 2019, 6 p.m. (UTC) - Feb. 14, 2020, 11:59 a.m. (AoE)

view all competition news

The competition has ended. To view your score and the score of the best teams, please refer to the ranking tab.

Problem Description | Ranking | Instance Format

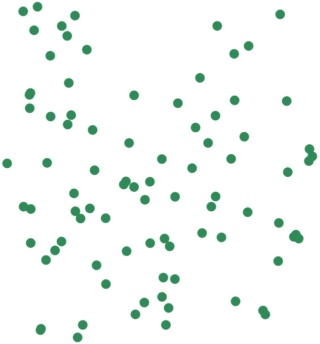
Minimum Convex Partition Problem

We are happy to announce the CG Challenge 2020, as part of CG Week in Zurich, Switzerland, June 22-26, 2020. As in the CG Challenge 2019, the objective will be to compute good solutions to instances of a difficult geometric optimization problem.

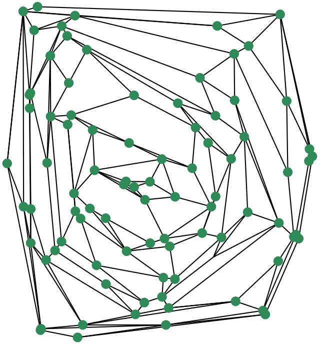
The contributors with the best solutions will be recognized at CG Week and invited to present their results. In addition, the top contributing teams will be invited to submit an extended abstract describing their methods and results, to be included in the LIPIcs proceedings of SoCG.

3-Approximation

- The 3APX tool implements the algorithm by Knauer and Spillner 2006.

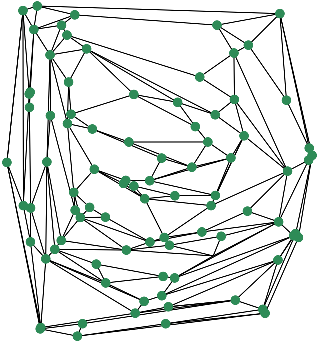


- The 3APX tool implements the algorithm by Knauer and Spillner 2006.



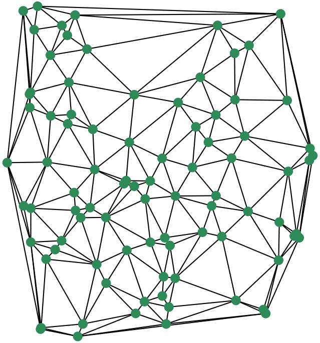
3-Approximation

- The 3APX tool implements the algorithm by Knauer and Spillner 2006.
- We extended 3APX by an approach based on onion layers.
- The decompositions generated contained lots of extremely long and thin triangles.

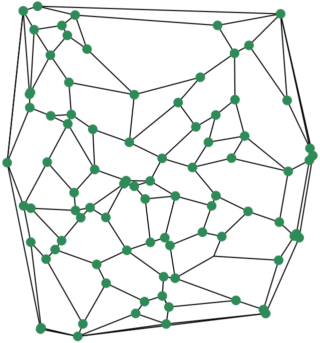


Merging Triangles

- ***Simple idea:*** Start with a Delaunay triangulation ...

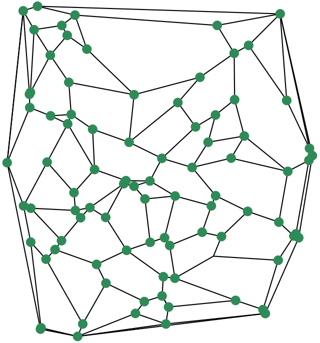


- ***Simple idea:*** Start with a Delaunay triangulation and merge neighboring faces.



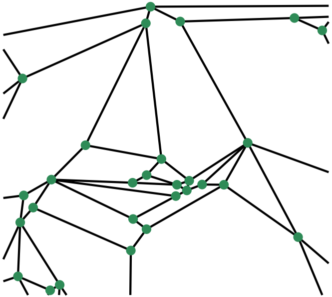
Merging Triangles

- ***Simple idea:*** Start with a Delaunay triangulation and merge neighboring faces.
- Our first implementation MERGEREFINE easily beat 3APX.
- This initial success motivated the development of a more sophisticated strategy.

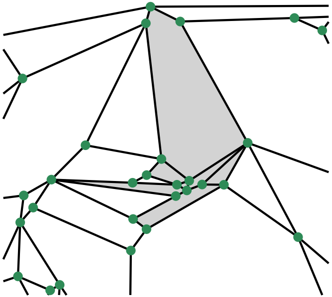


An Improved Implementation

- RECURSOR introduces several heuristics.

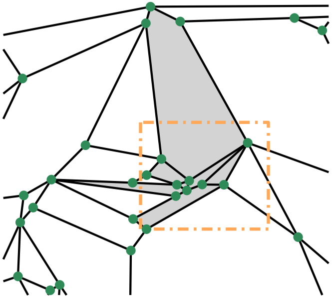


- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.



An Improved Implementation

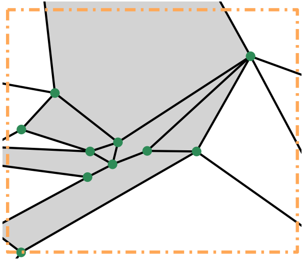
- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.



UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

An Improved Implementation

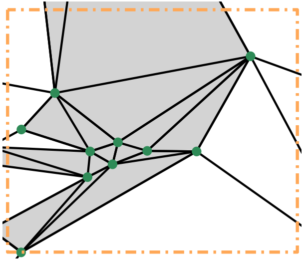
- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.



UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

An Improved Implementation

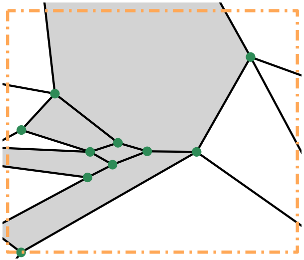
- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.



UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

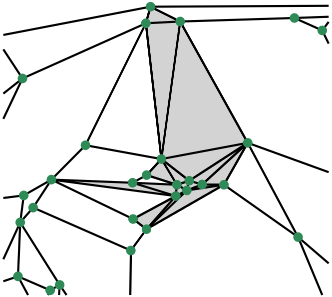
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.



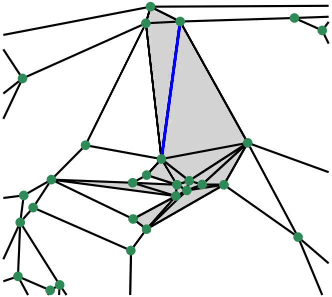
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.
- ***Edge flips***: Perform random edge flips on the triangulation of a hole.



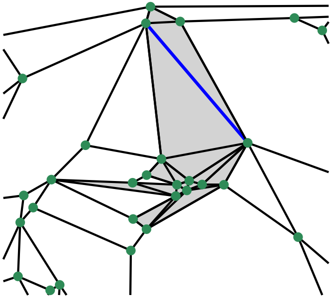
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.
- ***Edge flips***: Perform random edge flips on the triangulation of a hole.



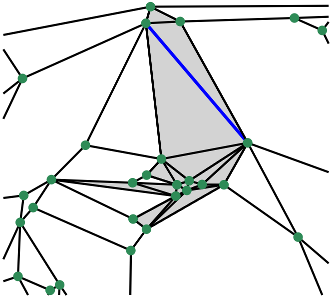
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.
- ***Edge flips***: Perform random edge flips on the triangulation of a hole.



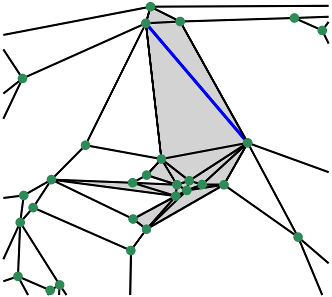
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.
- ***Edge flips***: Perform random edge flips on the triangulation of a hole.
- ***Continuous refinement***: Load a previous decomposition and try to improve it.



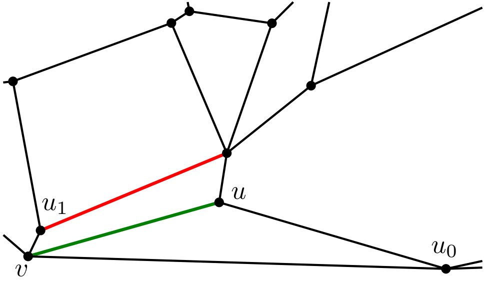
An Improved Implementation

- RECURSOR introduces several heuristics.
- ***Hole refinement***: Re-triangulate holes in a decomposition. Drop the newly inserted edges randomly without violating the convexity of the faces.
- ***Edge flips***: Perform random edge flips on the triangulation of a hole.
- ***Continuous refinement***: Load a previous decomposition and try to improve it.
- ***Parallel recursor***: Partition a decomposition into several non-overlapping sets of faces.



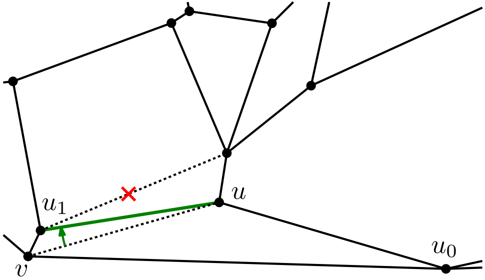
Flipping Edges

- FLIPPER was implemented relatively late.
- It picks a high degree vertex and rotates incident edges.
- Unnecessary edges are removed.
- FLIPPER interacts with RECURSOR as it re-structures the respective decompositions.



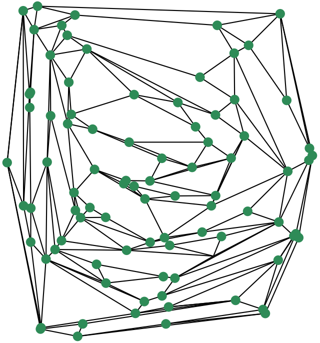
Flipping Edges

- FLIPPER was implemented relatively late.
- It picks a high degree vertex and rotates incident edges.
- Unnecessary edges are removed.
- FLIPPER interacts with RECURSOR as it re-structures the respective decompositions.



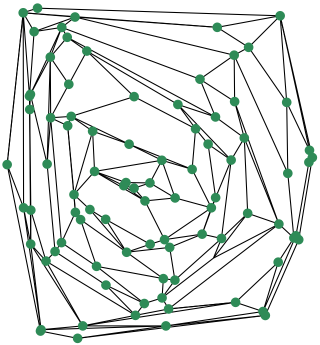
3APX (random)

#Faces 111



3APX (random) → 3APX (onion)

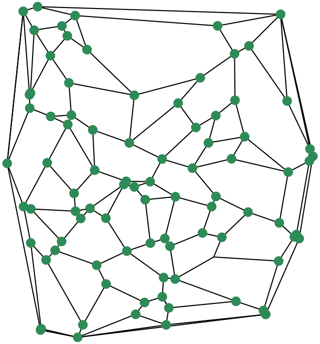
#Faces 100



Example Decomposition

3APX (random) → 3APX (onion) → MERGEREFINE

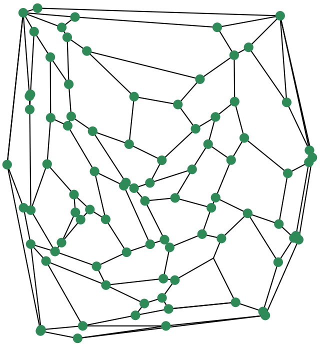
#Faces 63



Example Decomposition

3APX (random) → 3APX (onion) → MERGEREFINE → RECURSOR + FLIPPER

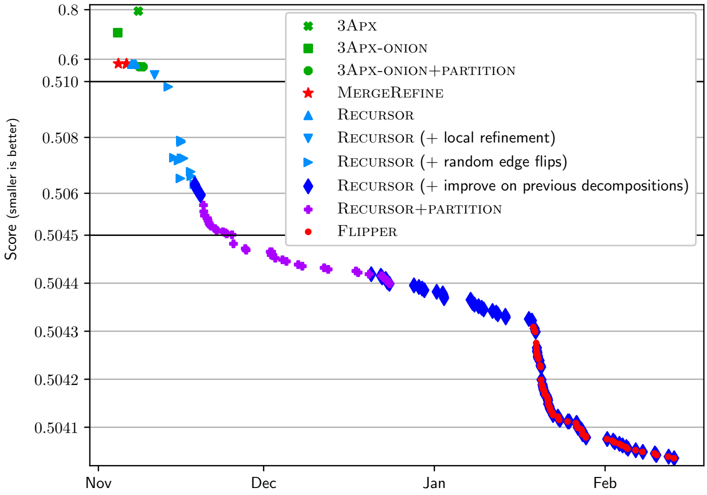
#Faces 54



Score Over Time

- We ran our tools on a wide variety of different computers.
- The estimated quality of a given decomposition is based on its ***score***.

$$\text{score} := \frac{\text{number of edges in convex partition}}{\text{number of edges in triangulation}}$$



Skip

Overview

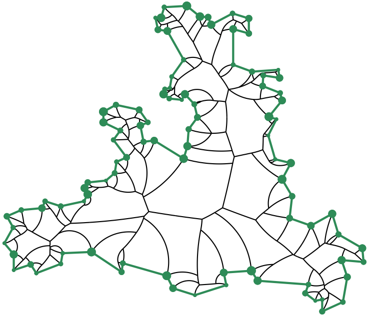
An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

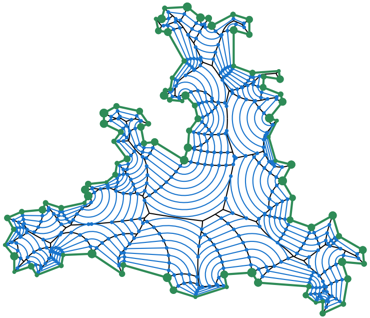
On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*



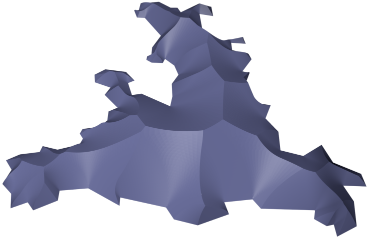
- The Voronoi regions of the generalized weighted Voronoi Diagrams are possibly disconnected.

Variable-Radius Skeleton



- The Voronoi regions of the generalized weighted Voronoi Diagrams are possibly disconnected.
- Thus, we introduce the variable-radius skeleton.
- It has a linear combinatorial complexity as its region stay connected in any case.

Variable-Radius Skeleton



- The Voronoi regions of the generalized weighted Voronoi Diagrams are possibly disconnected.
- Thus, we introduce the variable-radius skeleton.
- It has a linear combinatorial complexity as its region stay connected in any case.
- ***Variable-radius roofs*** can be derived from variable-radius skeletons.

Overview

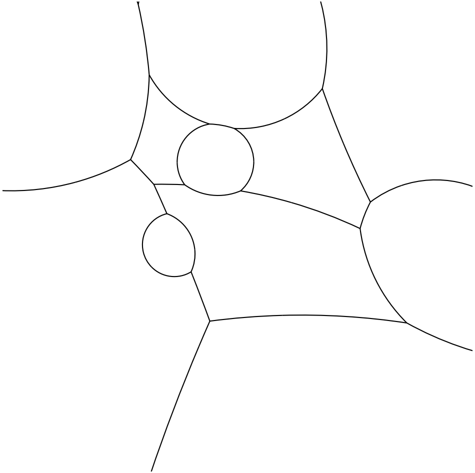
An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

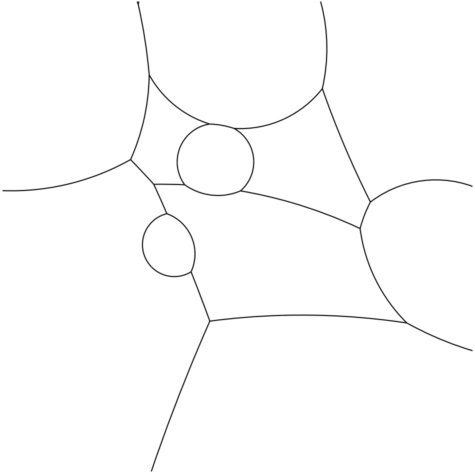
On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*



Definition (Weighted Bisector Graph)

A weighted bisector graph is a geometric graph whose faces are bounded by arcs that are portions of multiplicatively weighted bisectors of pairs of (point) sites such that each of its faces is defined by exactly one site.

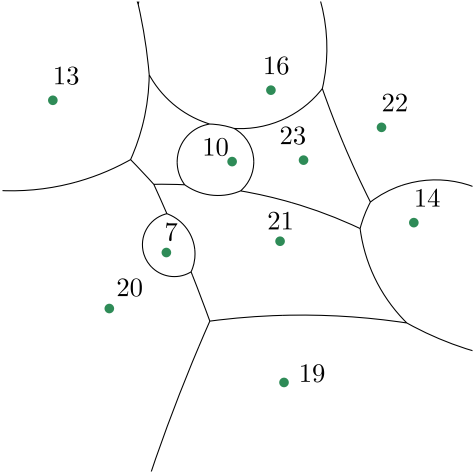


Definition (Weighted Bisector Graph)

A weighted bisector graph is a geometric graph whose faces are bounded by arcs that are portions of multiplicatively weighted bisectors of pairs of (point) sites such that each of its faces is defined by exactly one site.

Problem

Given: A partition G of the plane into faces.

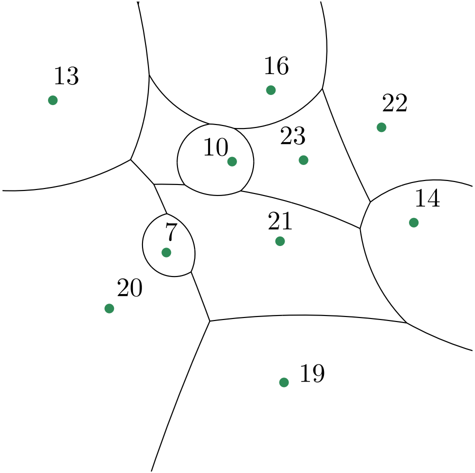


Definition (Weighted Bisector Graph)

A weighted bisector graph is a geometric graph whose faces are bounded by arcs that are portions of multiplicatively weighted bisectors of pairs of (point) sites such that each of its faces is defined by exactly one site.

Problem

Given: A partition G of the plane into faces.
Find: A set of points and suitable weights such that G is a bisector graph of the weighted points, if a solution exists.



Definition (Weighted Bisector Graph)

A weighted bisector graph is a geometric graph whose faces are bounded by arcs that are portions of multiplicatively weighted bisectors of pairs of (point) sites such that each of its faces is defined by exactly one site.

Problem

Given: A partition G of the plane into faces.
Find: A set of points and suitable weights such that G is a bisector graph of the weighted points, if a solution exists.

Theorem (Eder, Held, de Lorenzo, and Palfrader 2021)

If G is a graph that is regular of degree three then we can decide in $\mathcal{O}(m)$ time whether it is a bisector graph, where m denotes the combinatorial complexity of G.

Overview

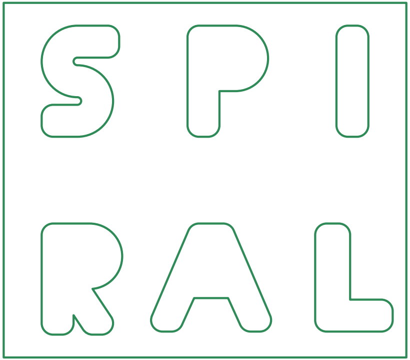
An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams
HELD AND DE LORENZO
Published in *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*

Weighted Skeletal Structures for Computing Variable-Radius Offsets
HELD AND DE LORENZO
Published in *Computer-Aided Design and Applications (CAD&A 2021)*

On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs
EDER, HELD, DE LORENZO, AND PALFRADER
Submitted to *Computational Geometry: Theory and Applications*

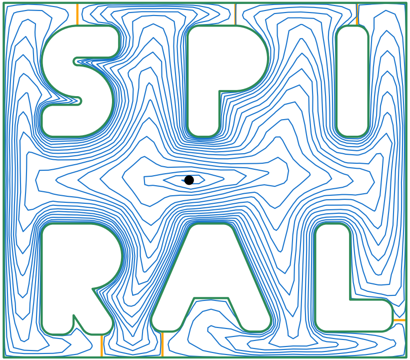
On the Generation of Spiral-Like Paths Within Planar Shapes
HELD AND DE LORENZO
Published in *Journal of Computational Design and Engineering (JCDE 2018)*

Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions
EDER, HELD, DE LORENZO, AND PALFRADER
Published in *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*



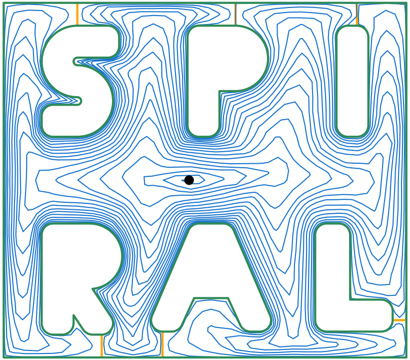
Problem

Given: A planar shape P that is bounded by straight-line segments and circular arcs as well as a ***step-over*** value $\delta > 0$.



Problem

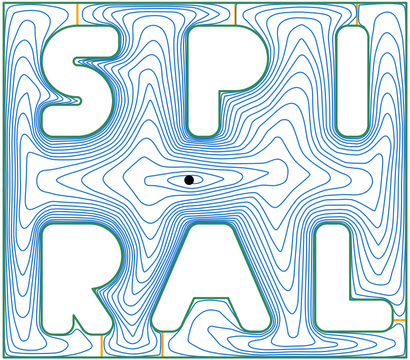
Given: A planar shape P that is bounded by straight-line segments and circular arcs as well as a ***step-over*** value $\delta > 0$.
Find: A spiral path that (1) consists of straight-line segments, (2) has no self-intersections, (3) starts in the interior and ends at the boundary of the shape, and (4) respects the user-specified maximum step-over distance δ.



Problem

Given: A planar shape P that is bounded by straight-line segments and circular arcs as well as a ***step-over*** value $\delta > 0$.
Find: A spiral path that (1) consists of straight-line segments, (2) has no self-intersections, (3) starts in the interior and ends at the boundary of the shape, and (4) respects the user-specified maximum step-over distance δ.

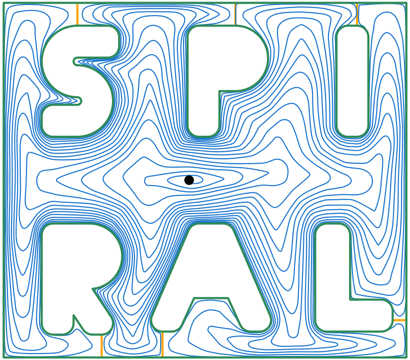
- Our approach is based on the medial axis within P.



Problem

Given: A planar shape P that is bounded by straight-line segments and circular arcs as well as a ***step-over*** value $\delta > 0$.
Find: A spiral path that (1) consists of straight-line segments, (2) has no self-intersections, (3) starts in the interior and ends at the boundary of the shape, and (4) respects the user-specified maximum step-over distance δ.

- Our approach is based on the medial axis within P.
- For ***high-speed machining***, it is important to avoid sharp corners.
- Thus, we smooth the spiral path using ***cubic B-splines***.

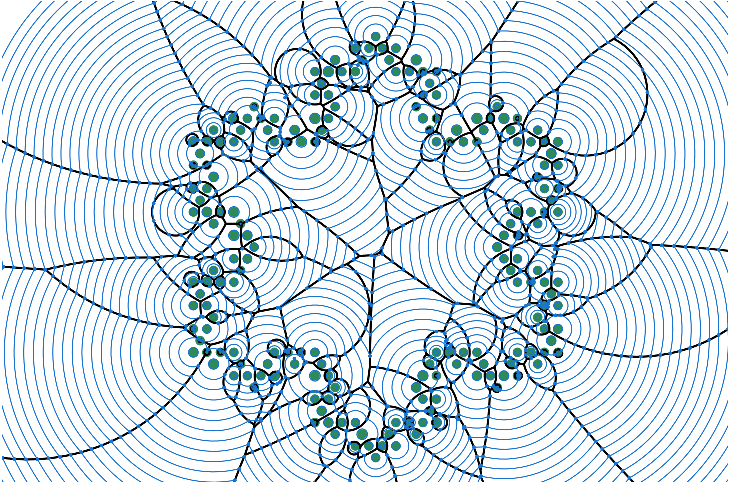


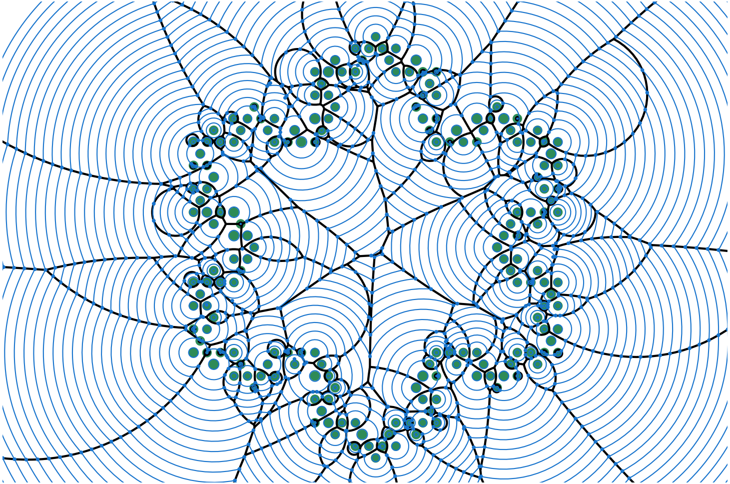
Problem

Given: A planar shape P that is bounded by straight-line segments and circular arcs as well as a ***step-over*** value $\delta > 0$.
Find: A spiral path that (1) consists of straight-line segments, (2) has no self-intersections, (3) starts in the interior and ends at the boundary of the shape, and (4) respects the user-specified maximum step-over distance δ.

- Our approach is based on the medial axis within P.
- For ***high-speed machining***, it is important to avoid sharp corners.
- Thus, we smooth the spiral path using ***cubic B-splines***.
- Additionally, it is also possible to generate ***double spirals***.

The End





Thank you for your attention!

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

References I

- Aurenhammer, Franz and Herbert Edelsbrunner (1984). "An Optimal Algorithm for Constructing the Weighted Voronoi Diagram in the Plane". In: ***Pattern Recognition*** 17.2, pp. 251–257. DOI: `10.1016/0031-3203(84)90064-5`.
- Eder, Günther, Martin Held, Stefan de Lorenzo, and Peter Palfrader (June 2020). "Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions". In: ***Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)***. Vol. 164. Leibniz International Proceedings in Informatics (LIPIcs). Zürich, Switzerland: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 85:1–85:10. ISBN: 978-3-95977-143-6. DOI: `10.4230/LIPIcs.SoCG.2020.85`.
- – (Feb. 2021). "On the Recognition and Reconstruction of Weighted Voronoi Diagrams and Bisector Graphs". Submitted to *Computational Geometry: Theory and Applications*.
- Fortune, Steven (1987). "A Sweepline Algorithm for Voronoi Diagrams". In: ***Algorithmica*** 2.1-4, p. 153. DOI: `10.1007/BF01840357`.
- Har-Peled, Sariel and Benjamin Raichel (2015). "On the Complexity of Randomly Weighted Multiplicative Voronoi Diagrams". In: ***Discrete & Computational Geometry*** 53.3, pp. 547–568. DOI: `10.1007/s00454-015-9675-0`.
- Held, Martin and Stefan de Lorenzo (July 2018). "On the Generation of Spiral-Like Paths Within Planar Shapes". In: ***Journal of Computational Design and Engineering (JCDE 2018)*** 5.3, pp. 348–357. DOI: `10.1016/j.jcde.2017.11.011`.

References II

Held, Martin and Stefan de Lorenzo (Aug. 2020). “An Efficient, Practical Algorithm and Implementation for Computing Multiplicatively Weighted Voronoi Diagrams”. In: ***Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)***. Vol. 173. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 56:1–56:15. ISBN: 978-3-95977-162-7. DOI: 10.4230/LIPIcs.ESA.2020.56.

– (Jan. 2021). “Weighted Skeletal Structures for Computing Variable-Radius Offsets”. In: ***Computer-Aided Design and Applications (CAD&A 2021)*** 18.5, pp. 875–889. DOI: 10.14733/cadaps.2021.875-889.

Held, Martin and Stefan Huber (2009). “Topology-Oriented Incremental Computation of Voronoi Diagrams of Circular Arcs and Straight-Line Segments”. In: ***Computer-Aided Design*** 41.5, pp. 327–338. DOI: 10.1016/j.cad.2008.08.004.

Hillewaert, Hans (2010). ***Giraffa Camelopardalis Rothschildi***. URL: https://commons.wikimedia.org/wiki/File:Giraffa_camelopardalis_rothschildi_(pattern).jpg.

Kaplan, Haim, Edgar Ramos, and Micha Sharir (2011). “The Overlay of Minimization Diagrams in a Randomized Incremental Construction”. In: ***Discrete & Computational Geometry*** 45.3, pp. 371–382. DOI: 10.1007/s00454-010-9324-6.

Keats, Derek (2009). ***Coral Patterns Closer***. URL: https://commons.wikimedia.org/wiki/File:Coral_patterns_closer_(6163706598).jpg.

Knauer, Christian and Andreas Spillner (2006). "Approximation Algorithms for the Minimum Convex Partition Problem". In: ***Scandinavian Workshop on Algorithm Theory***. Springer, pp. 232–241. DOI: 10.1007/11785293_23.

Rader, Matthew T (2009). ***A Western Honey Bee on a Honeycomb***. URL: https://commons.wikimedia.org/wiki/File:Western_honey_bee_on_a_honeycomb.jpg.

Shamos, Michael Ian and Dan Hoey (1975). "Closest-Point Problems". In: ***Foundations of Computer Science, 1975., 16th Annual Symposium on***. IEEE, pp. 151–162. DOI: 10.1109/SFCS.1975.8.

Yap, Chee-Keng (1987). "An $\mathcal{O}(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments". In: ***Discrete & Computational Geometry*** 2.4, pp. 365–393. DOI: 10.1007/BF02187890.