Computing Low-Cost Convex Partitions for Planar Point Sets Based on Tailored Decompositions

Günther Eder Martin Held Stefan de Lorenzo Peter Palfrader

University of Salzburg, Department of Computer Science, Salzburg, Austria

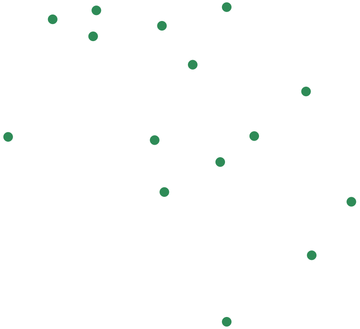
June 24, 2020

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

Problem Specification

Problem

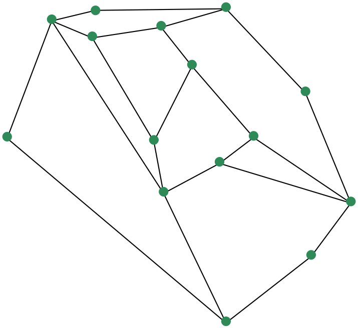
Given: A set P of n points in the plane.



Problem Specification

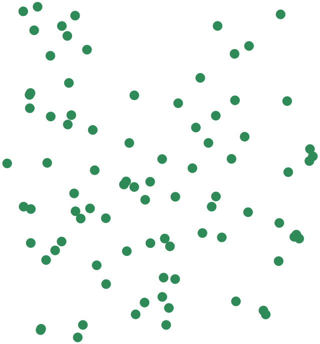
Problem

Given: A set P of n points in the plane.
Compute: A plane graph with vertex set P (with each point in P having positive degree) that partitions the convex hull of P 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.



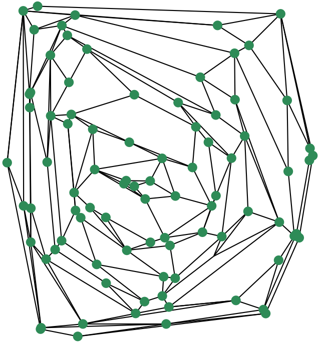
3-Approximation

- The 3APX tool implements the algorithm by Knauer and Spillner [KS06].



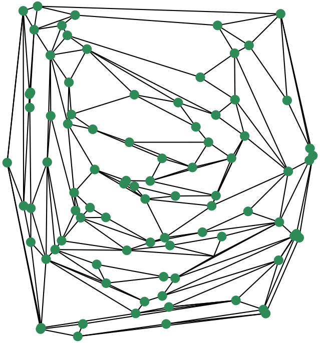
3-Approximation

- The 3APX tool implements the algorithm by Knauer and Spillner [KS06].



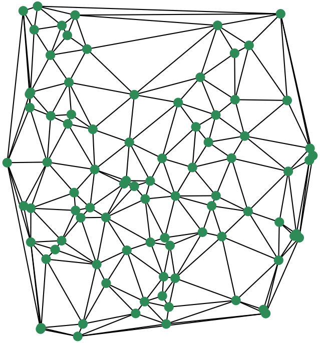
3-Approximation

- The 3APX tool implements the algorithm by Knauer and Spillner [KS06].
- We extended 3APX by an approach based on onion layers.
- The decompositions generated contained lots of extremely long and thin triangles.
- A partitioning into smaller “cells” did not reduce the face counts substantially.



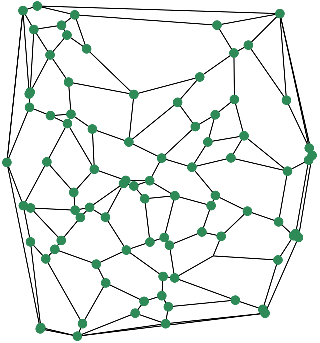
Merging Triangles

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



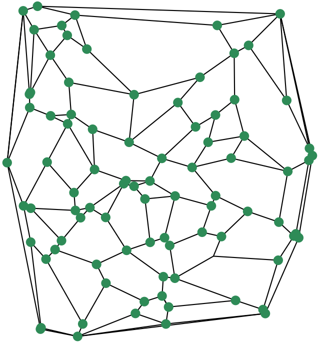
Merging Triangles

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



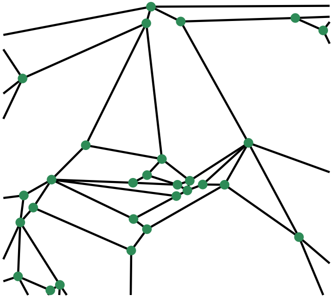
Merging Triangles

- **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.



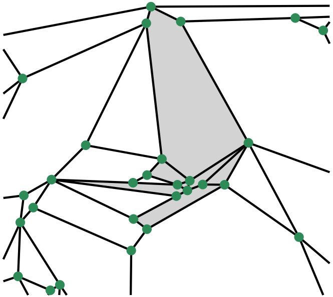
An Improved Implementation

- RECURSOR introduces several heuristics.



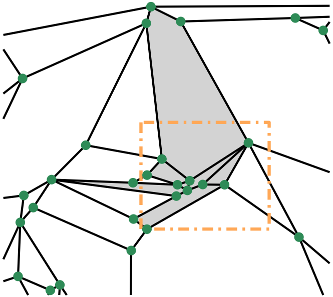
An Improved Implementation

- 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.



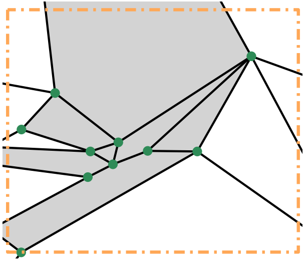
An Improved Implementation

- 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.



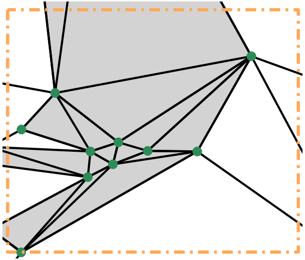
An Improved Implementation

- 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.



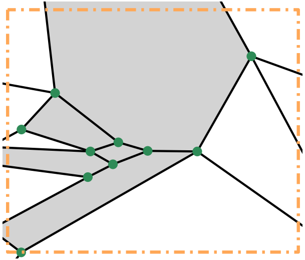
An Improved Implementation

- 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.



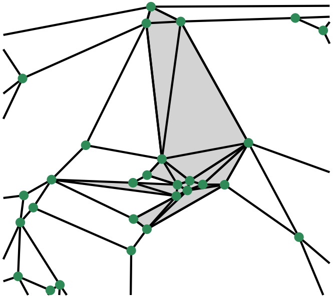
An Improved Implementation

- 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.



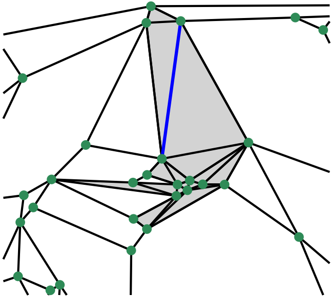
An Improved Implementation

- 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.



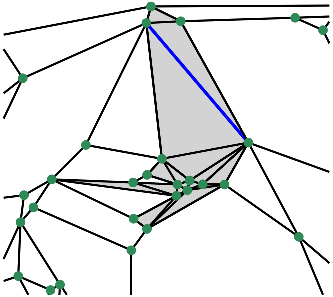
An Improved Implementation

- 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.



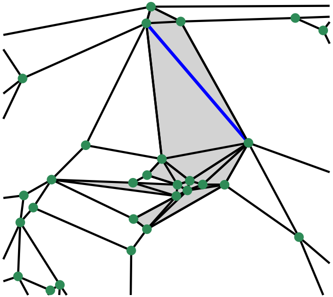
An Improved Implementation

- 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.



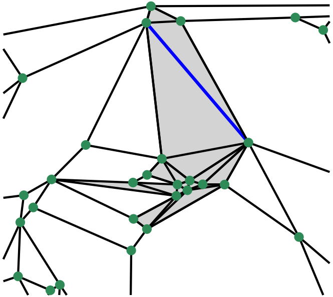
An Improved Implementation

- 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.



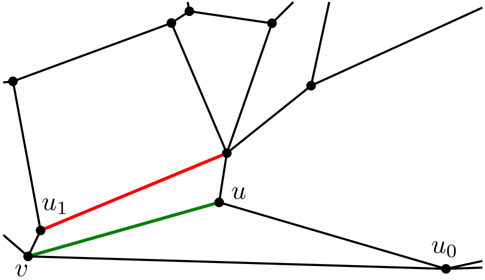
An Improved Implementation

- 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.



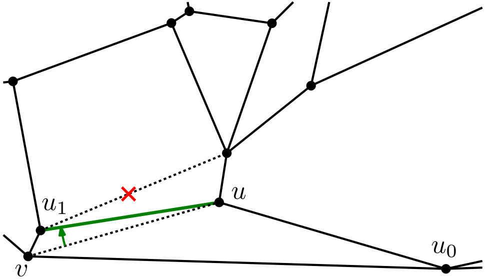
Flipping Edges

- 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

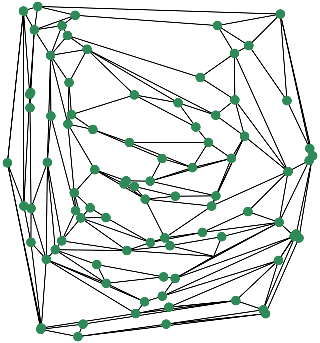
- FLIPPER was implemented relatively late.
- It picks a high degree vertex and rotates incident edges.
- Unnecessary edges are removed.
- FLIPPER interacts with RECURSOR as it re-structures the respective decompositions.



Example Decomposition

3APX (random)

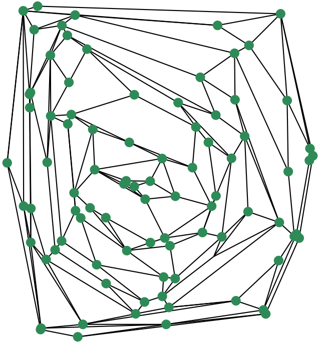
#Faces 111



Example Decomposition

3APX (random) → 3APX (onion)

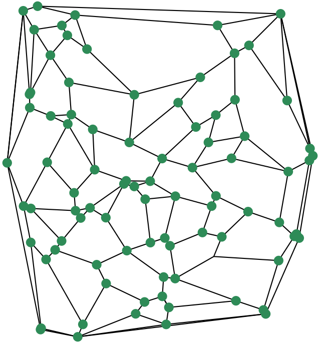
#Faces 100



Example Decomposition

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

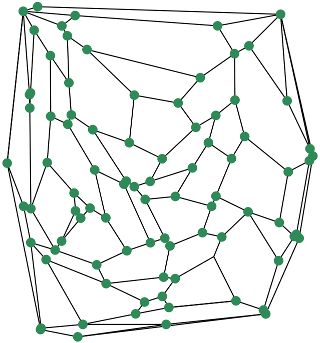
#Faces 63



Example Decomposition

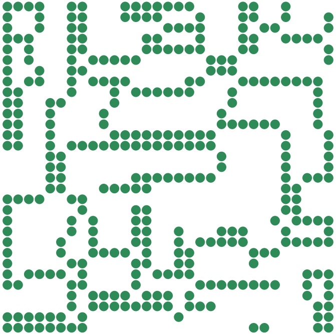
3APX (random) → 3APX (onion) → MERGEREFINE → RECURSOR + FLIPPER

#Faces 54



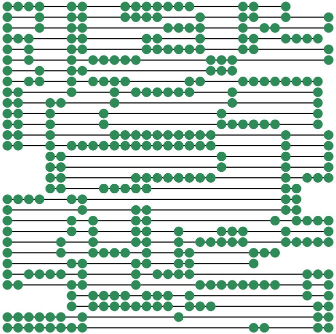
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.



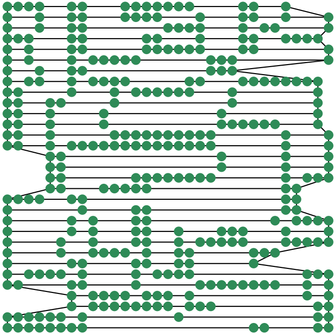
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.
- ORTHOOPT generates initial decompositions that are geared towards this type of input sets.



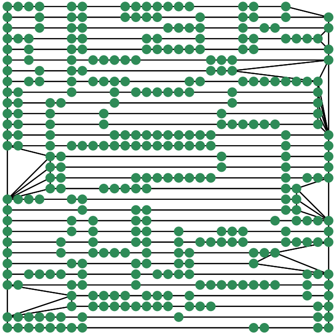
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.
- ORTHOOPT generates initial decompositions that are geared towards this type of input sets.



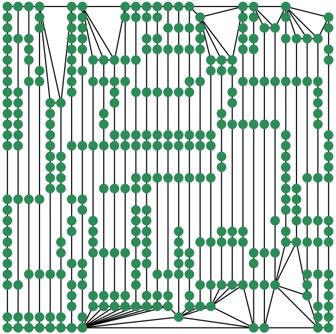
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.
- ORTHOOPT generates initial decompositions that are geared towards this type of input sets.



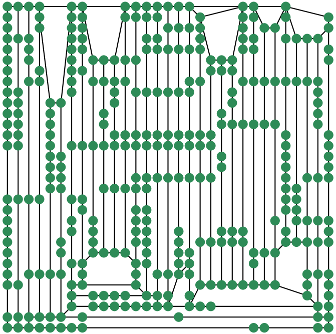
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.
- ORTHOOPT generates initial decompositions that are geared towards this type of input sets.



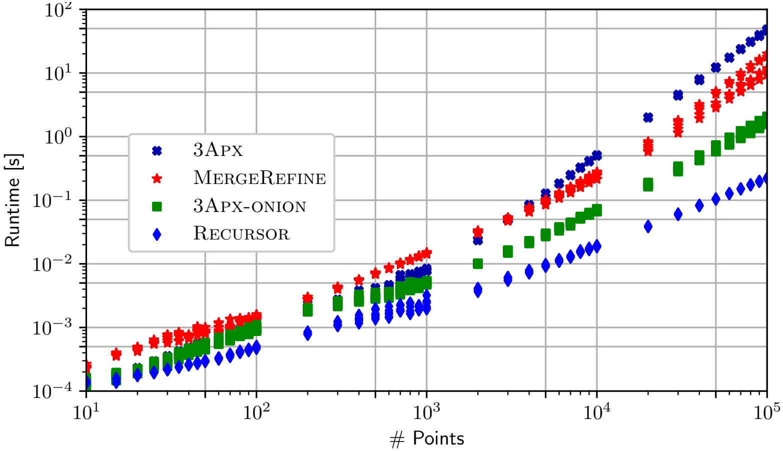
Orthogonal Optimizer

- Later on a second batch of instances was made available.
- These instances contained points sampled from a dense integer grid.
- OrthoOpt generates initial decompositions that are geared towards this type of input sets.
- These initial decompositions were improved by Flipper and Recursor.



Experimental Results

- All runtime tests were carried out on an Intel Core i7-6700 CPU clocked at 3.40 GHz.



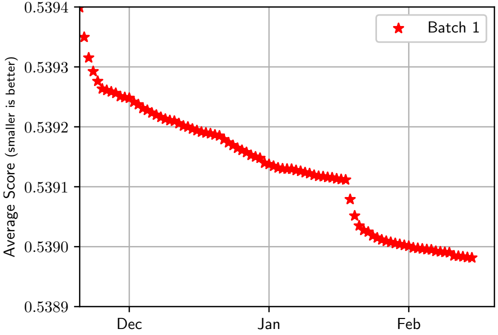
Average Score Over Time

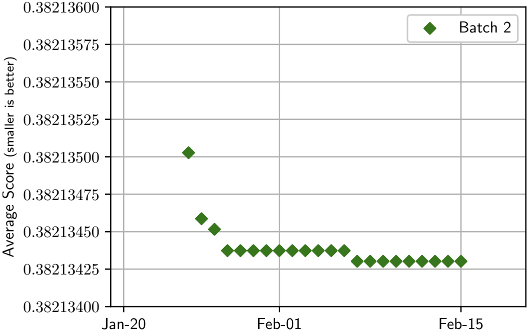
- We ran our tools on a wide variety of different computers.

Average Score Over Time

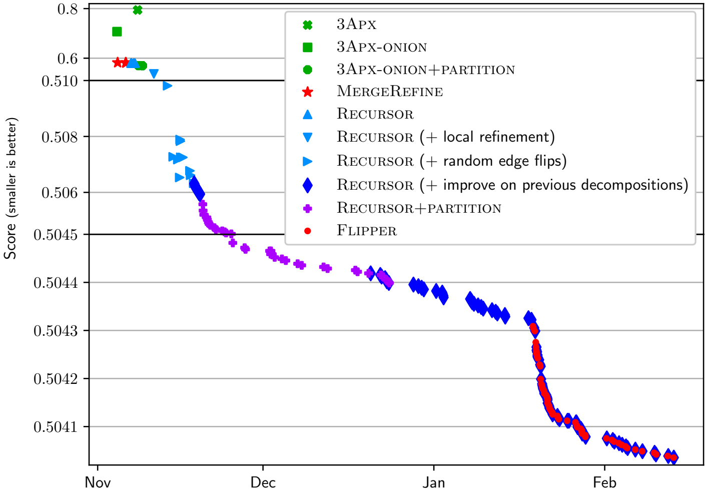
- We ran our tools on a wide variety of different computers.
- The estimated quality of a given decomposition is based on its ***score***.

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

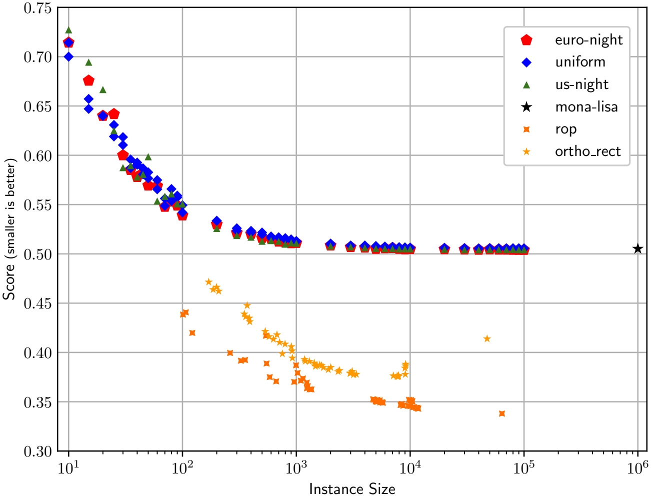




Score Over Time



Score Per Instance



Thank You For Your Attention!

Günther Eder

Martin Held

Stefan de Lorenzo

Peter Palfrader

UNIVERSITÄT SALZBURG
Computational Geometry and Applications Lab

References I

Christian Knauer and Andreas Spillner.
Approximation Algorithms for the Minimum Convex Partition Problem.
In *Algorithm Theory – SWAT 2006*, pages 232–241, 2006.