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
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 $\pi$. 
• The 3APX tool implements the algorithm by Knauer and Spillner [KS06].
• The 3APX tool implements the algorithm by Knauer and Spillner [KS06].
• 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.
• **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.

- **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.
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.
• **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

$3A_{\text{PX}}$ (random)

#Faces 111
Example Decomposition

$3A\text{PX (random)} \rightarrow 3A\text{PX (onion)}$

#Faces 100
Example Decomposition

$3\text{APX (random)} \rightarrow 3\text{APX (onion)} \rightarrow \text{MERGE\textsc{Refine}}$

#Faces 63
Example Decomposition

3APX (random) $\rightarrow$ 3APX (onion) $\rightarrow$ MERGEREFINE $\rightarrow$ RECURSOR + FLIPPER

#Faces 54
• Later on a second batch of instances was made available.
• These instances contained points sampled from a dense integer grid.
• 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.
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.
• 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.
• 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.
• 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.
All runtime tests were carried out on an Intel Core i7-6700 CPU clocked at 3.40 GHz.
We ran our tools on a wide variety of different computers.
• We ran our tools on a wide variety of different computers.
• The estimated quality of a given decomposition is based on its **score**.

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

---

**Computational Geometry and Applications Lab**

**UNIVERSITÄT SALZBURG**

Günther Eder  
Martin Held  
Stefan de Lorenzo  
Peter Palfrader
Score Per Instance

![Graph showing the relationship between Score (smaller is better) and Instance Size for various instances: euro-night, uniform, us-night, mona-lisa, rop, ortho_rect.]
Thank You For Your Attention!