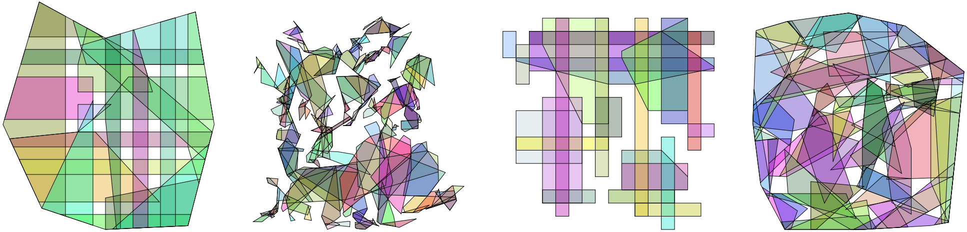
Shadoks Approach to Convex Covering

Guilherme D. da Fonseca – LIS, Aix-Marseille Université

CG:SHOP 2023



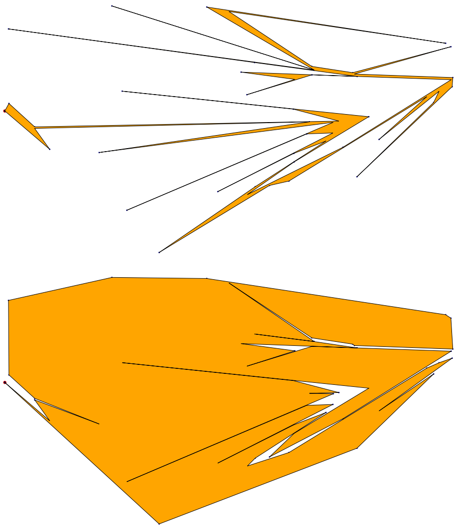
CG:SHOP Competition

- Part of SoCG (International Symposium on Computational Geometry)
- 5th year, started in 2018–2019
- Hard geometric optimization problems
- Different problem each year
- ~ 200 instances given
- ~ 3 months to compute solutions
- Send our solutions (not the code)
- Score based on the quality of the solutions
- Top teams invited to publish in SoCG proceedings and ACM Journal of Experimental Algorithmics
- This talk is about the 2023 competition, but let's look at other years...

Minimum (or Maximum) Area Polygon:

- Input: A set of points $S \subset \mathbb{R}^2$
- Output: A simple polygon with vertex set S
- Goal: Minimize (or maximize) the area

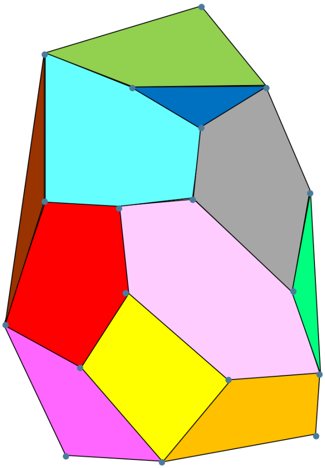
- Related to Euclidean TSP
- Two categories: minimization, maximization
- We got 2nd place
- Techniques: greedy and local search



Minimum Convex Partition:

- Input: A set of points $S \subset \mathbb{R}^2$
- Output: A simple partition of the convex hull of S into convex regions with vertex set S
- Goal: Minimize the number of regions

- We got 4th place
- Used integer programming



11 convex regions

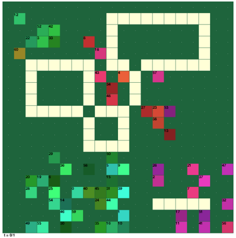
CG:SHOP 2021

Coordinated Motion Planning:

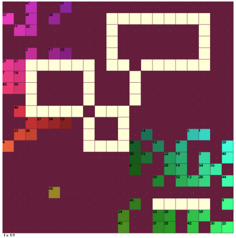
- Input: Sets $S, T \subset \mathbb{Z}^2$ of start and target locations for n robots and possibly a set of obstacles
- Output: A sequence of movements for all robots from start to target avoiding collisions
- Goal: Minimize the total time (makespan) or the total number of movements (energy)

- 1st place in makespan category,
  3rd place in energy category
- Used storage network and conflict optimizer

Start:



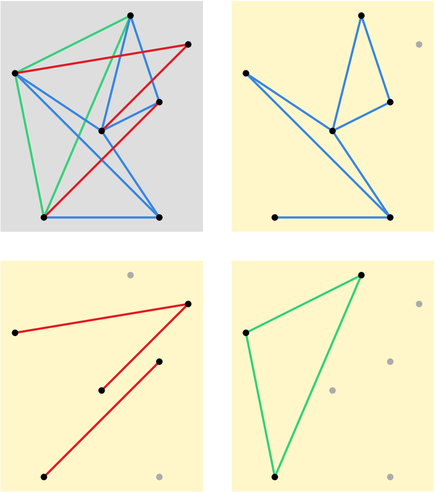
Target:



Partition Into Plane Graphs:

- Input: A graph G embedded in the plane with straight edges
- Output: A partition of G into plane graphs
- Goal: Minimize the number of partitions (colors)

- We won 1st place
- Best solution of all teams to all instances
- Optimal solution to at least 23
- Reused conflict optimizer



Knapsack Translational Packing:

- Input: A convex polygon (*container*) and a multi-set of polygons with *values* (*items*)
- Output: A translation of some items that form a packing inside the container
- Goal: Maximize the sum of the values in the output

- We won 1st place
- Used greedy, local search, and integer programming

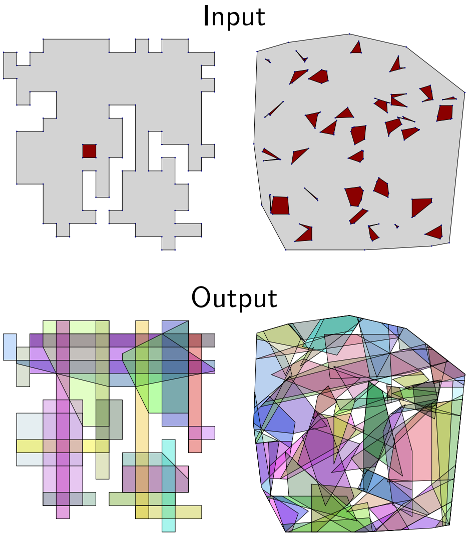
Shadoks

Introduction
Competition
Problems
2023
Approach
Phase 1
Bron-Kerbosch
Vertices
Bloating
Phase 2
Witnesses
Set Cover
Bootstrapping
Questions
Implementation
Thanks

Convex Covering:

- Input: A polygon with holes P
- Output: A collection of convex polygons whose union is P
- Goal: Minimize the number of convex polygons

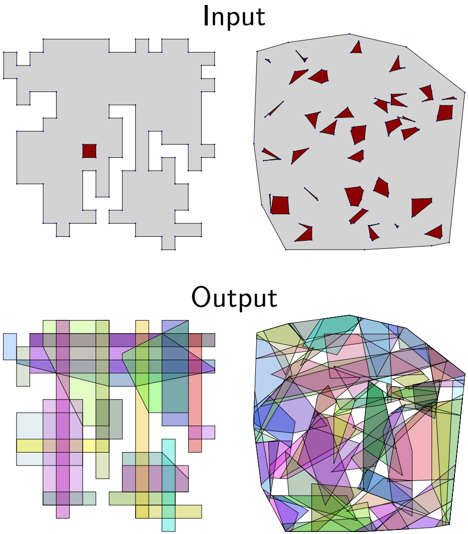
- We won 2nd place
- Best solution among all teams to 128 of 206 instances



Convex Covering:

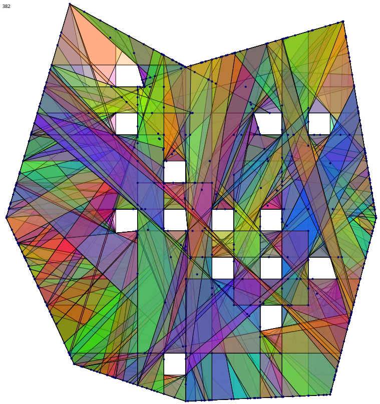
- Input: A polygon with holes P
- Output: A collection of convex polygons whose union is P
- Goal: Minimize the number of convex polygons

- We won 2nd place
- Best solution among all teams to 128 of 206 instances

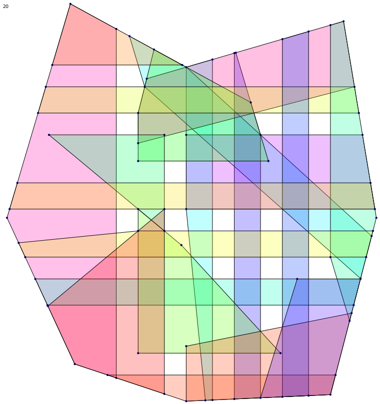


Two-Phase Approach

1. Find many large convex polygons inside P
2. Cover P with few of them



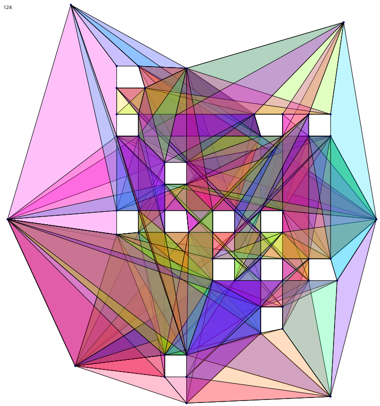
382 convex polygons $\longrightarrow$



20 convex polygons

Phase 1: Build a Collection

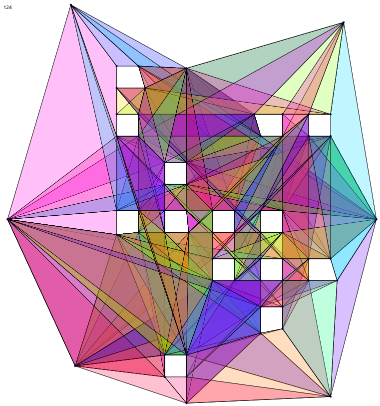
- Build a collection $\mathcal{C}$ of convex polygons:
  - Polygons inside P
  - Union covers P
  - Contains a small subset $\mathcal{S} \subseteq \mathcal{C}$ that covers P
  - Convex polygons are large
  - There are many of them (but not too many)



124 convex polygons

Phase 1: Build a Collection

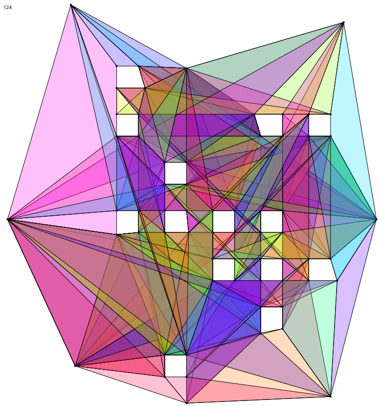
- Build a collection $\mathcal{C}$ of convex polygons:
  - Polygons inside P
  - Union covers P
  - Contains a small subset $\mathcal{S} \subseteq \mathcal{C}$ that covers P
  - Convex polygons are large
  - There are many of them (but not too many)



124 convex polygons

Phase 1: Build a Collection

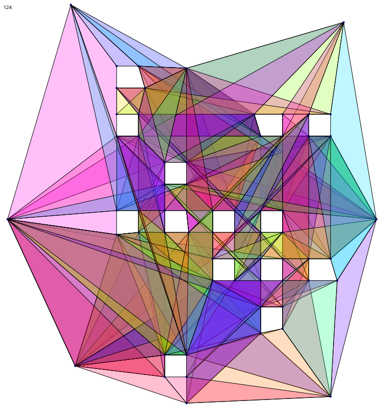
- Build a collection $\mathcal{C}$ of convex polygons:
  - Polygons inside P
  - Union covers P
  - Contains a small subset $\mathcal{S} \subseteq \mathcal{C}$ that covers P
  - Convex polygons are large
  - There are many of them (but not too many)



124 convex polygons

Phase 1: Build a Collection

- Build a collection $\mathcal{C}$ of convex polygons:
  - Polygons inside P
  - Union covers P
  - ~~Contains a small subset $\mathcal{S} \subseteq \mathcal{C}$ that covers P~~
  - Convex polygons are large
  - There are many of them (but not too many)



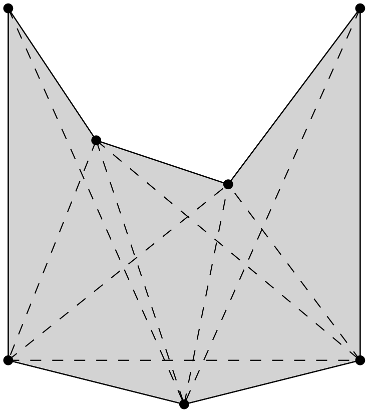
124 convex polygons

Bron-Kerbosch

V-Maximal convex polygon:

Convex polygon $C \subseteq P$ with vertices in V such that for all $p \in V \setminus C$, the convex hull of $C \cup \{p\}$ is not in P

- Classic practical algorithm to enumerate maximal cliques of a graph (V, E)
- In a polygon without holes, the maximal cliques in the visibility graph of V correspond to V-maximal sets
- Not true for polygons with holes
- Possible to extend Bron-Kerbosch

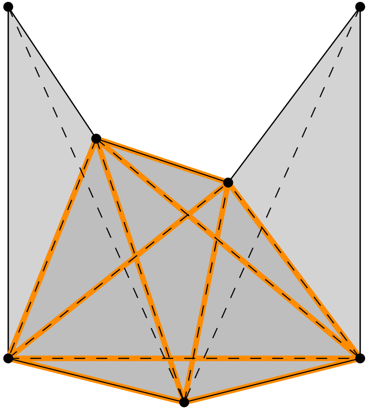


Bron-Kerbosch

V-Maximal convex polygon:

Convex polygon $C \subseteq P$ with vertices in V such that for all $p \in V \setminus C$, the convex hull of $C \cup \{p\}$ is not in P

- Classic practical algorithm to enumerate maximal cliques of a graph (V, E)
- In a polygon without holes, the maximal cliques in the visibility graph of V correspond to V-maximal sets
- Not true for polygons with holes
- Possible to extend Bron-Kerbosch

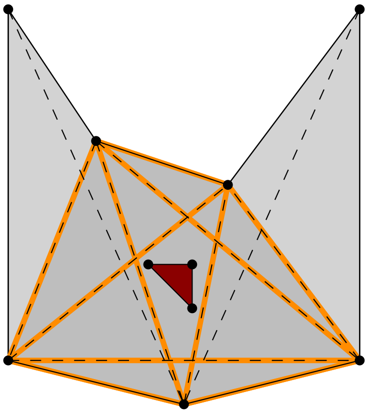


Bron-Kerbosch

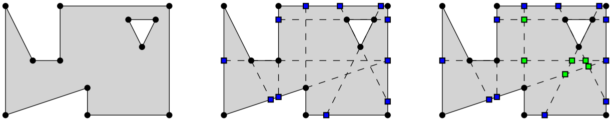
V-Maximal convex polygon:

Convex polygon $C \subseteq P$ with vertices in V such that for all $p \in V \setminus C$, the convex hull of $C \cup \{p\}$ is not in P

- Classic practical algorithm to enumerate maximal cliques of a graph (V, E)
- In a polygon without holes, the maximal cliques in the visibility graph of V correspond to V-maximal sets
- Not true for polygons with holes
- Possible to extend Bron-Kerbosch

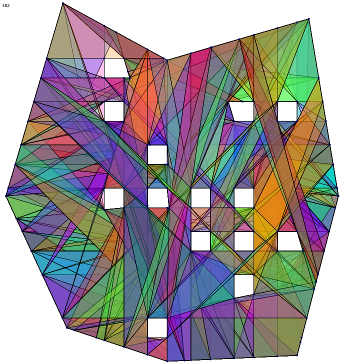


Vertices



V $\quad$ $V \cup S_1$ $\quad$ $V \cup S_2$

- Other points may be used, not only vertices
- Number of maximal polygons grows quickly
- Does not scale well
- Try another approach...



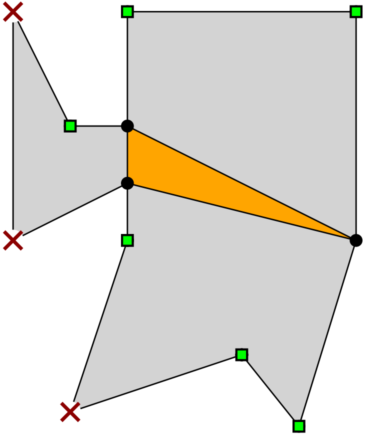
382 convex polygons for $V \cup S_1$

Random Bloating

C: convex polygon in P
S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\mathrm{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances

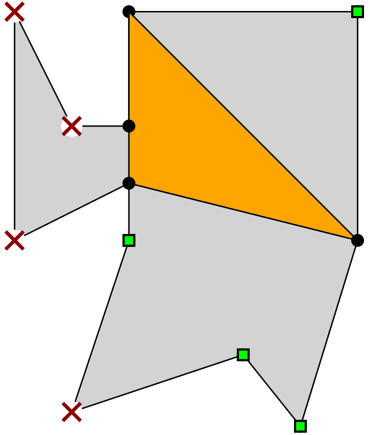


Random Bloating

C: convex polygon in P
S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\text{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances

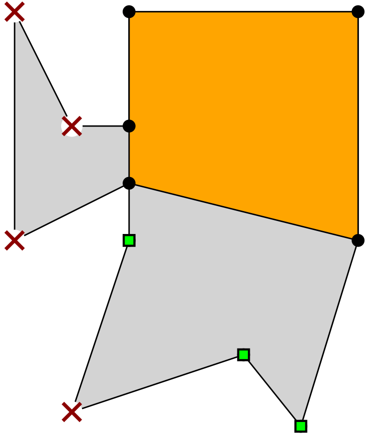


Random Bloating

C: convex polygon in P
S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\mathrm{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances

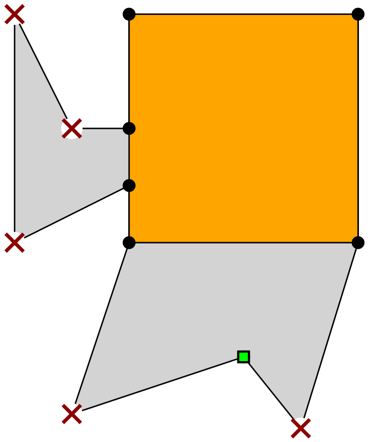


Random Bloating

C: convex polygon in P
S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\mathrm{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances



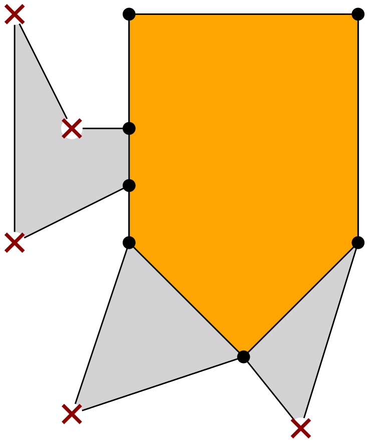
Random Bloating

C: convex polygon in P

S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\mathrm{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances

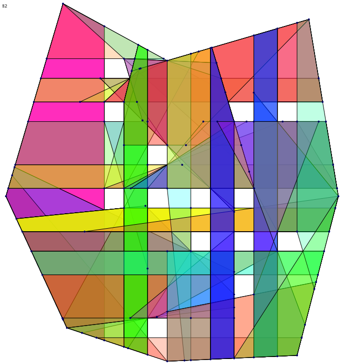


Random Bloating

C: convex polygon in P
S: set of points in P

1. Pick a random point $p \in S$
2. Remove p from S
3. If $\mathrm{conv}(C \cup \{p\}) \subseteq P$, then $C \leftarrow C \cup \{p\}$

- C from a constrained Delaunay triangulation or Bron-Kerbosch with $V \subset S$
- Much faster for large instances
- Slightly worse than Bron-Kerbosch for small instances

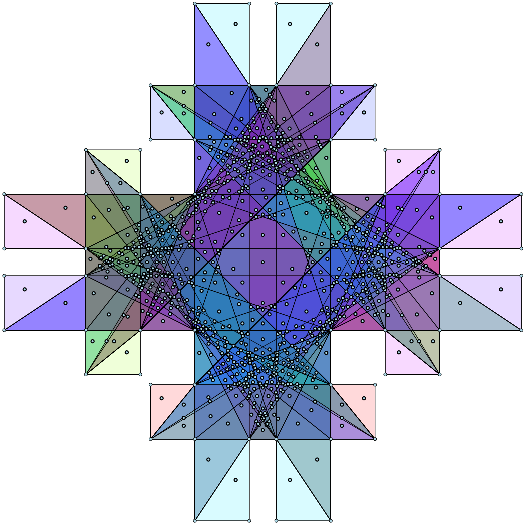


82 convex polygons from triangulation

Phase 2: Set Cover

$\mathcal{C}$: Convex polygons from phase 1
P: Instance polygon with holes

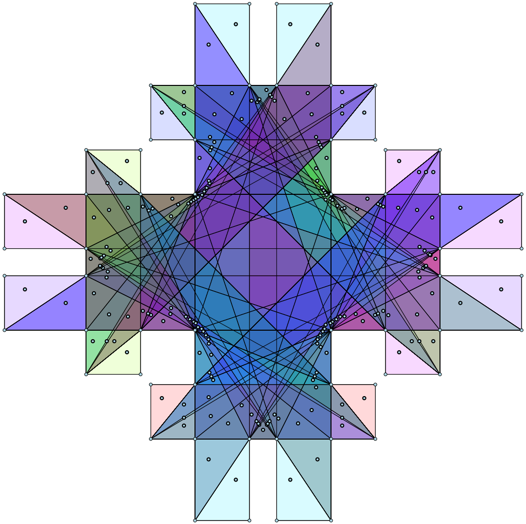
- $(\mathcal{C}, P)$ define a set system
- P has infinitely many points
- First attempt: reduce P to a quadratic number of witnesses, one point per arrangement cell
- Too many witnesses!
- Building the arrangement is slow!



1009 witnesses for 82 convex polygons

Vertex Witnesses

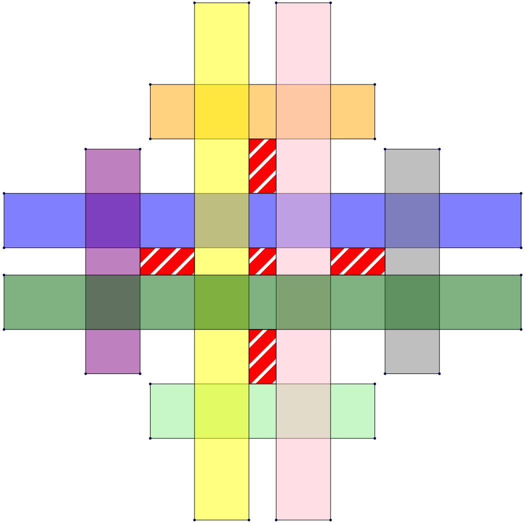
- Solution: only place witnesses near vertices of P
- Does not guarantee that P is covered
- Two possible fixes:
  - Add a witnesses inside each uncovered area and repeat (generally better, but slower)
  - Cover the uncovered area using some quick heuristic (faster and sometimes better)



200 witnesses for 82 convex polygons

Vertex Witnesses

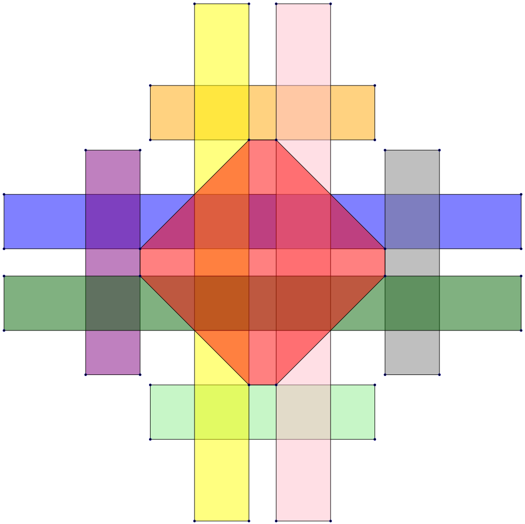
- Solution: only place witnesses near vertices of P
- Does not guarantee that P is covered
- Two possible fixes:
  - Add a witnesses inside each uncovered area and repeat (generally better, but slower)
  - Cover the uncovered area using some quick heuristic (faster and sometimes better)



5 uncovered regions 8 convex polygons

Vertex Witnesses

- Solution: only place witnesses near vertices of P
- Does not guarantee that P is covered
- Two possible fixes:
  - Add a witnesses inside each uncovered area and repeat (generally better, but slower)
  - Cover the uncovered area using some quick heuristic (faster and sometimes better)



0 uncovered regions 9 convex polygons

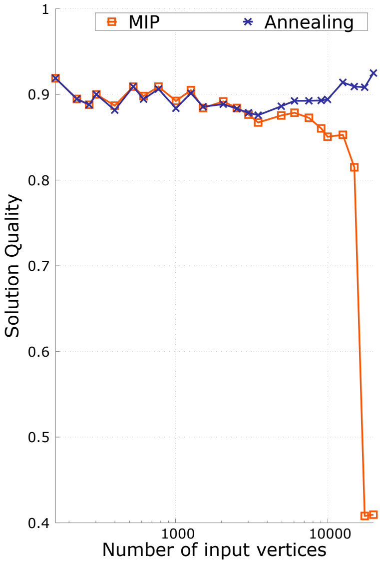
Solving Combinatorial Set Cover

Use mixed integer programming (MIP):

- Very fast for small to medium instances
- Solutions often guaranteed optimal
- On some large instances:
  slow and very bad solutions

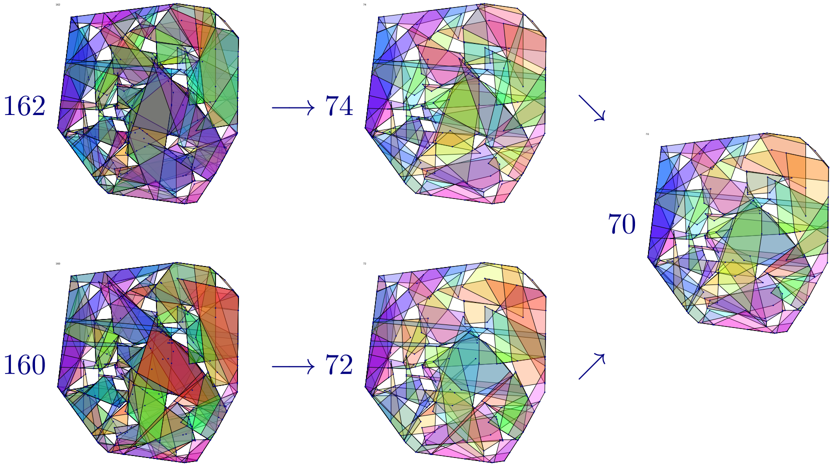
Use simulated annealing:

- Solutions close to optimal, but no guarantees
- Hard to decide how much time to wait before stopping
- Scales well to very large instances
- No need to use external libraries



Bootstrapping

Multiple good solutions can be combined into a collection and solved again



Questions

- The number of iterations when adding more witnesses is often very small
- Theoretical question:
  Is there a bound on the number of iterations using vertex witnesses?
- Theoretical question:
  Is a subquadratic number of witnesses always sufficient for a collection made of all V-maximal polygons?
- Theoretical question:
  Are there efficient enumeration algorithms for the V-maximal convex polygons?

Implementation

- Coded in C++ and compiled with gcc
- Executed on Fedora Linux using GNU Parallel
- Cplex for mixed integer programming
- Heavily uses **CGAL**:
  - Polygon union
  - Constrained Delaunay triangulation
  - Visibility graph
  - Arrangement
  - Convex hull

Thank You!

Art by Fanny Sanín