
APPLIED aspects of COMPUTATIONAL GEOMETRY

Minkowski Sums, The
General Polygonal Case

Dan Halperin
School of Computer Science
Tel Aviv University

Overview

- complexity
- algorithms
- practice: convex decomposition + union
 - good convex decompositions for Minkowski sums
 - handling degeneracies
- alternative methods
- offset polygons

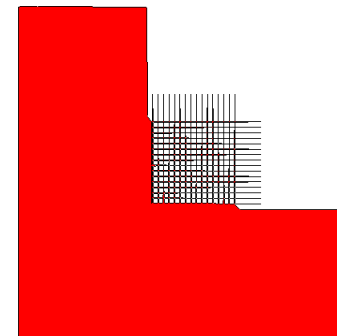
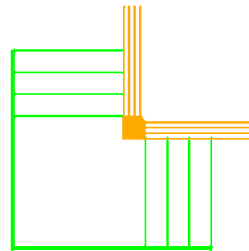
Reminder:

Minkowski sum of arbitrary polygonal sets

- $P_1 \oplus (P_2 \cup P_3) = (P_1 \oplus P_2) \cup (P_1 \oplus P_3)$
- Step 1 Decompose P and Q into convex subpolygons P_1, \dots, P_s and Q_1, \dots, Q_t
- Step 2 Compute $P_i \oplus Q_j$ for each pair
- Step 3 Construct the union of those subsums

Minkowski sum of arbitrary polygonal sets, complexity

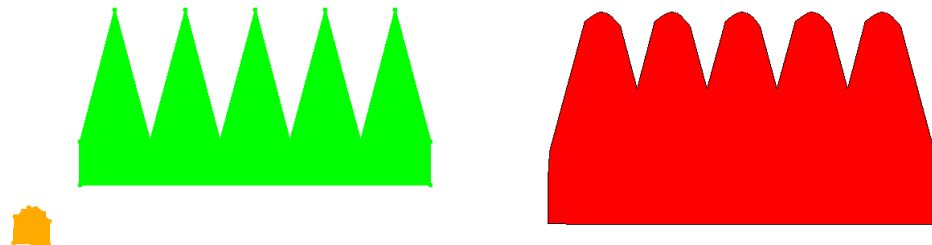
- two simple polygons with m, n vertices
- upper bound
contained in the arrangements of mn hexagons
hence $O(m^2n^2)$
- lower bound



Minkowski sum of polygonal sets: convex plus simple

- convex polygon with m vertices, simple polygon with n
- upper bound
 - pseudo-disc property
 - complexity of pseudo-disc convex polygons with a total of k vertices
 - in summary: $O(mn)$

- lower bound



The practice of the decomposition framework

- Step 1 Decompose P and Q into convex subpolygons P_1, \dots, P_s and Q_1, \dots, Q_t
- Step 2 Compute $R_{ij} := P_i \oplus Q_j$ for each pair
- Step 3 Construct the union of those subsums

Steps 1 and 3 are (were) challenging from a practical point of view. The main issues:

- union strategy
- handle degeneracies correctly in the union computation
- suitable decomposition

remarks:

- the oddity of computing the union of polygons; 3sum-hard problems
- running times in the experiments below obsolete, proportions remain

Step 3:

Constructing the union of the subsums

algorithms for computing the union of a set of convex polygons:

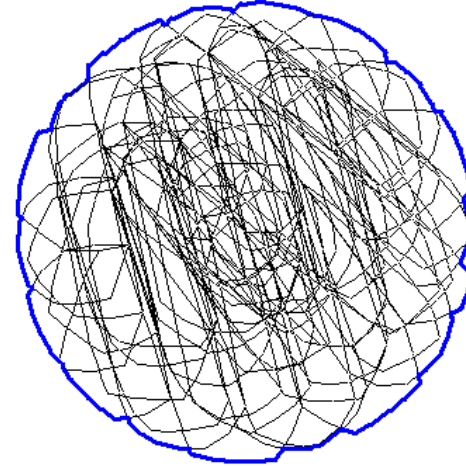
- **Arrangement** union algorithm
- **Incremental** union algorithm
- **Divide-and-Conquer** union algorithm

all algorithms handle degenerate inputs

recall that $R_{ij} := P_i \oplus Q_j$, and let $R = \cup\{R_{ij}\}$

Arrangement algorithm

- add all the edges of R into a planar arrangement



- compute carefully for each face, edge and vertex whether it is inside union
- time: $O((l+k) \log k)$ or $O(l+k \log k)$

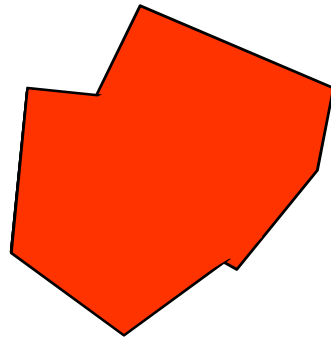
$O(l+k)$ - traversal

k - number of edges in R

l - number of intersections among edges of R

Incremental algorithm

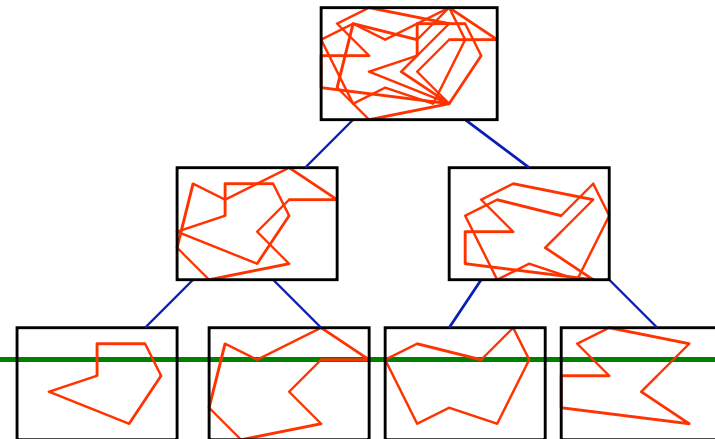
- add the polygons of R one after the other
- maintain the partial union as a planar map by removing redundant edges



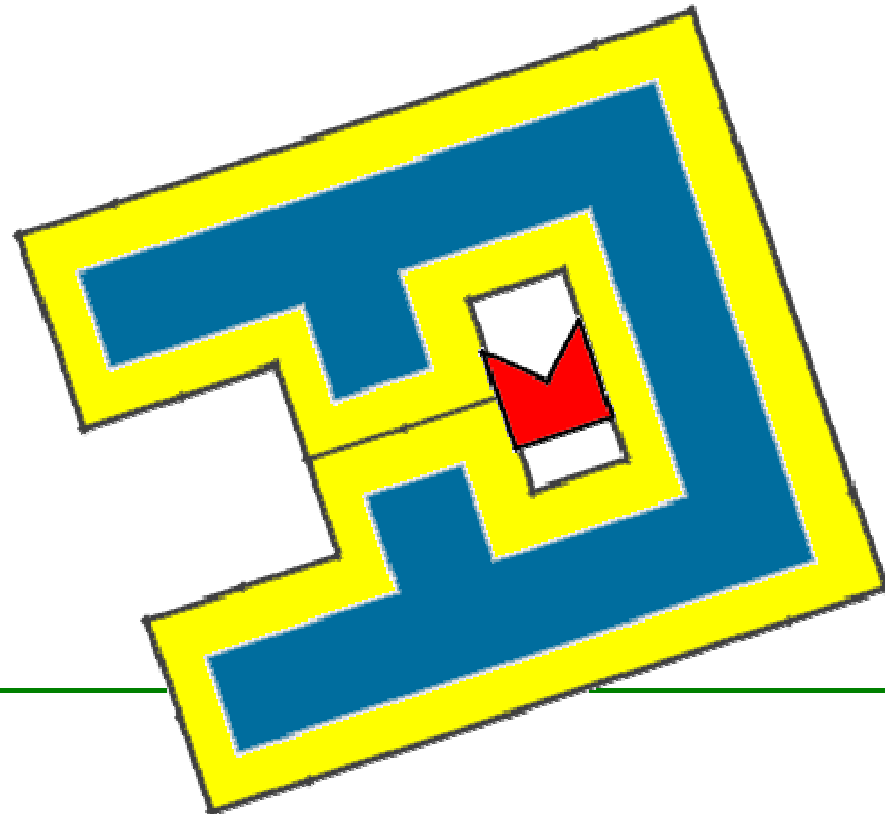
- practically works much better on most problems

Divide-and-conquer algorithm

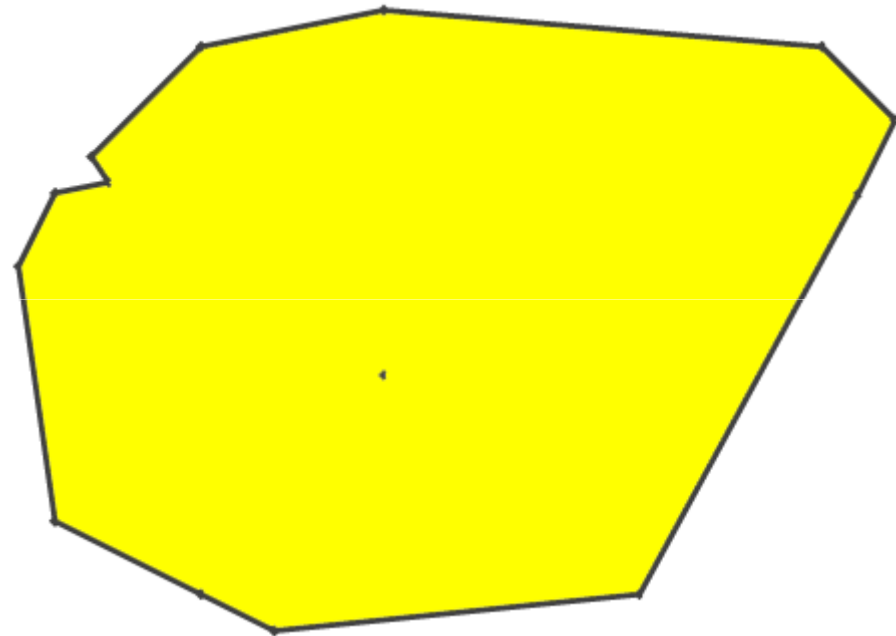
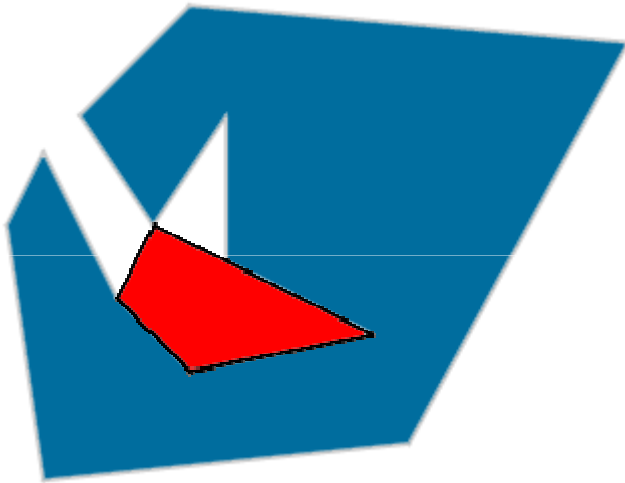
- using the incremental union algorithm compute the sum of P with every subpolygon of Q - the result are t maps
- using the Arrangement algorithm compute the union of each pair of maps to get $t/2$ maps
- repeat recursively $\log t$ times



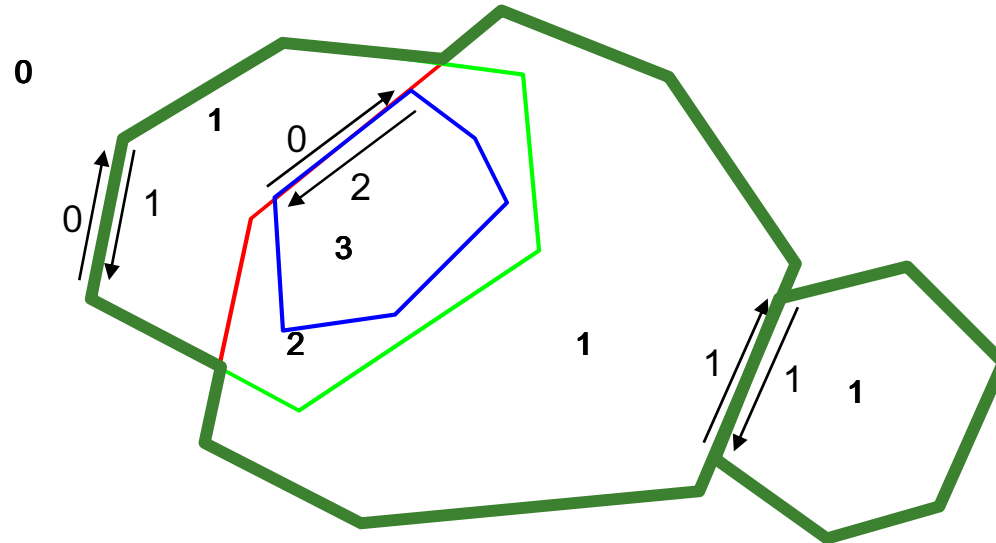
Degenerate case:
tight passage



Degenerate case:
tight placement



Arrangement union algorithm: Handling degeneracies

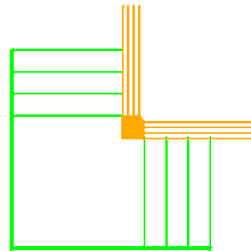


1. While inserting the polygons, maintain the *boundary count* for each halfedge
2. Update *inside count* for each face - in how many polygons it is contained
 $IC(f_2) = IC(f_1) - BC(e_1) + BC(e_2)$
3. Identify boundary edges by comparing the inside count to the boundary count

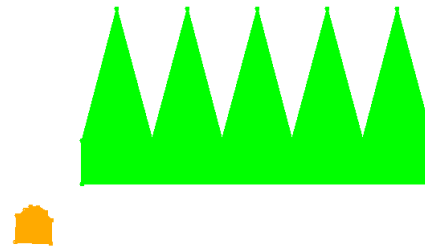
Sample input data



robot



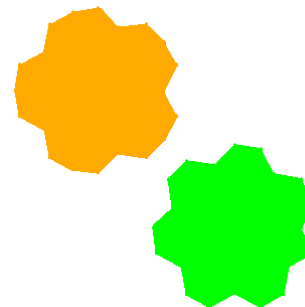
fork



comb

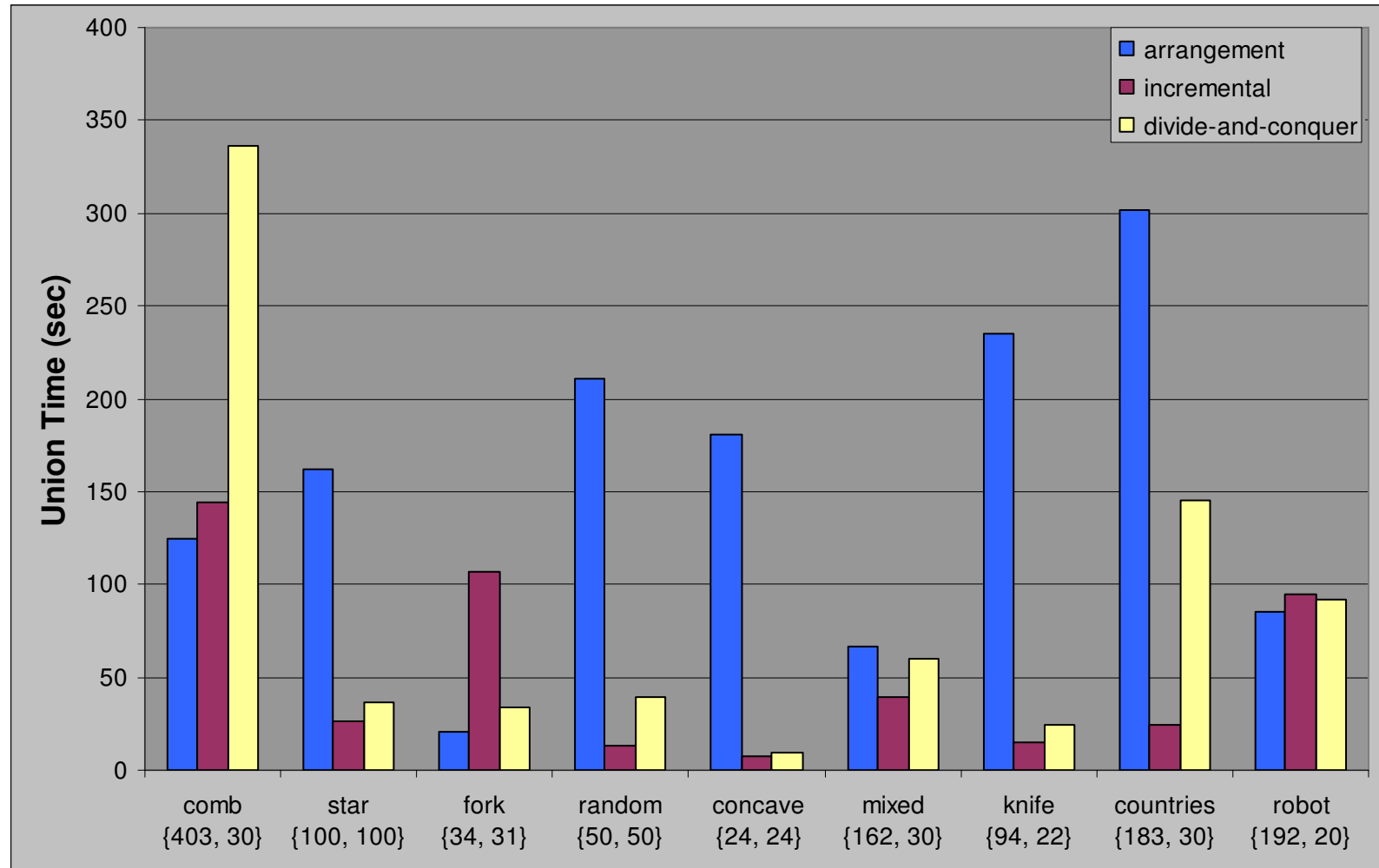


random

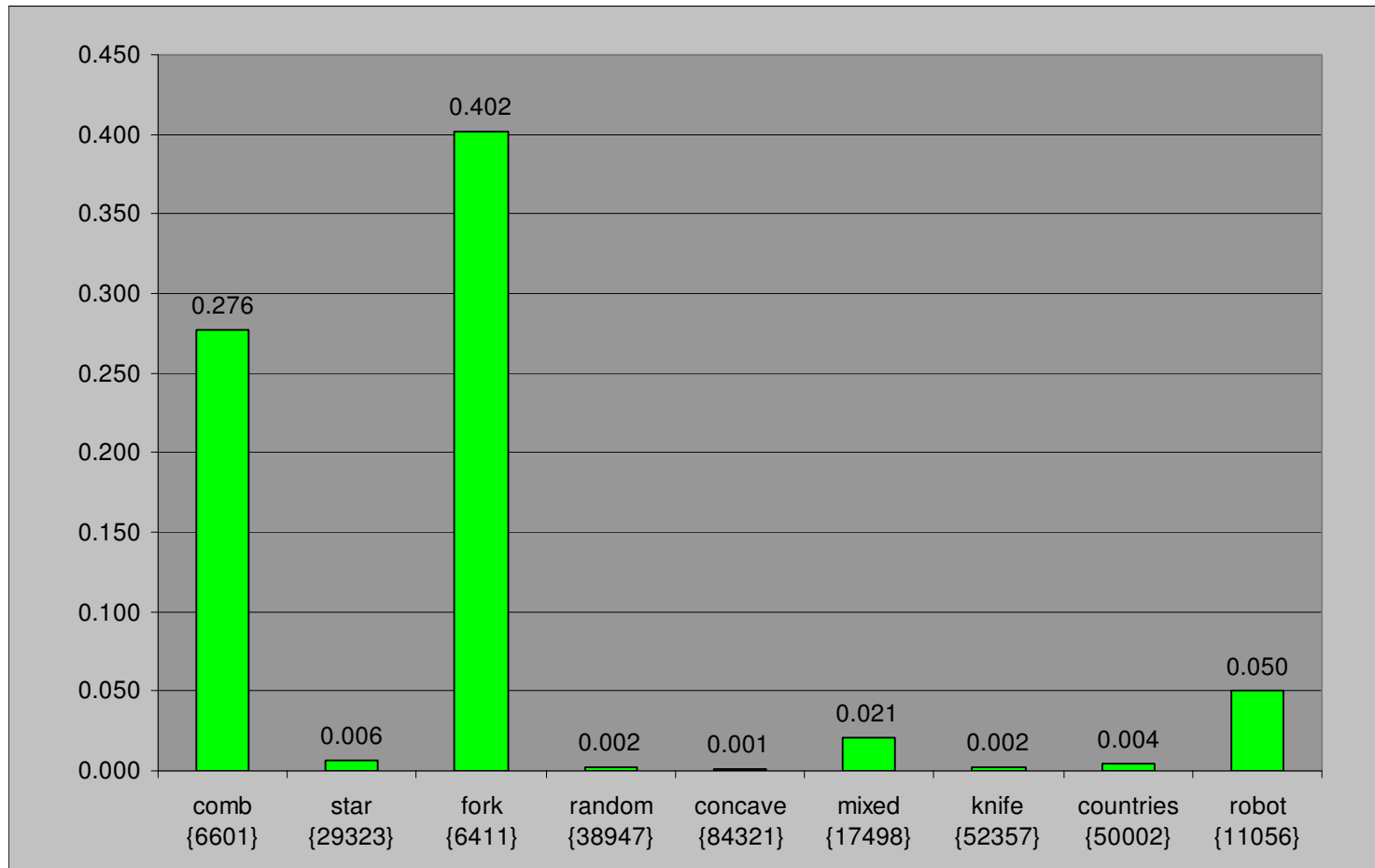


stars

Results: union construction time



Results: $C_{PQ} = M_{PQ}/V_{PQ}$



V_{PQ} - number of vertices in the underlying arrangement

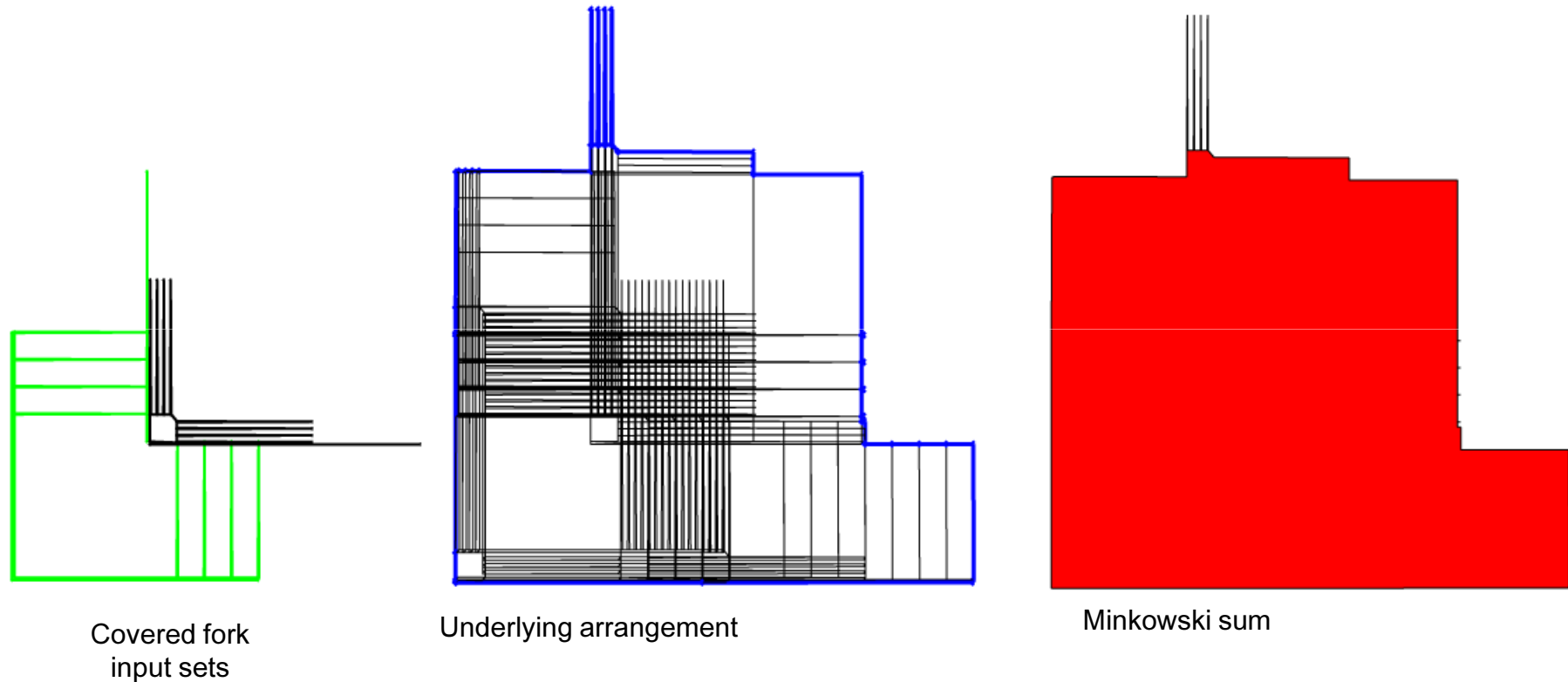
M_{PQ} - number of vertices on the boundary of $P \oplus Q$

Results:

Union time vs. C_{PQ}

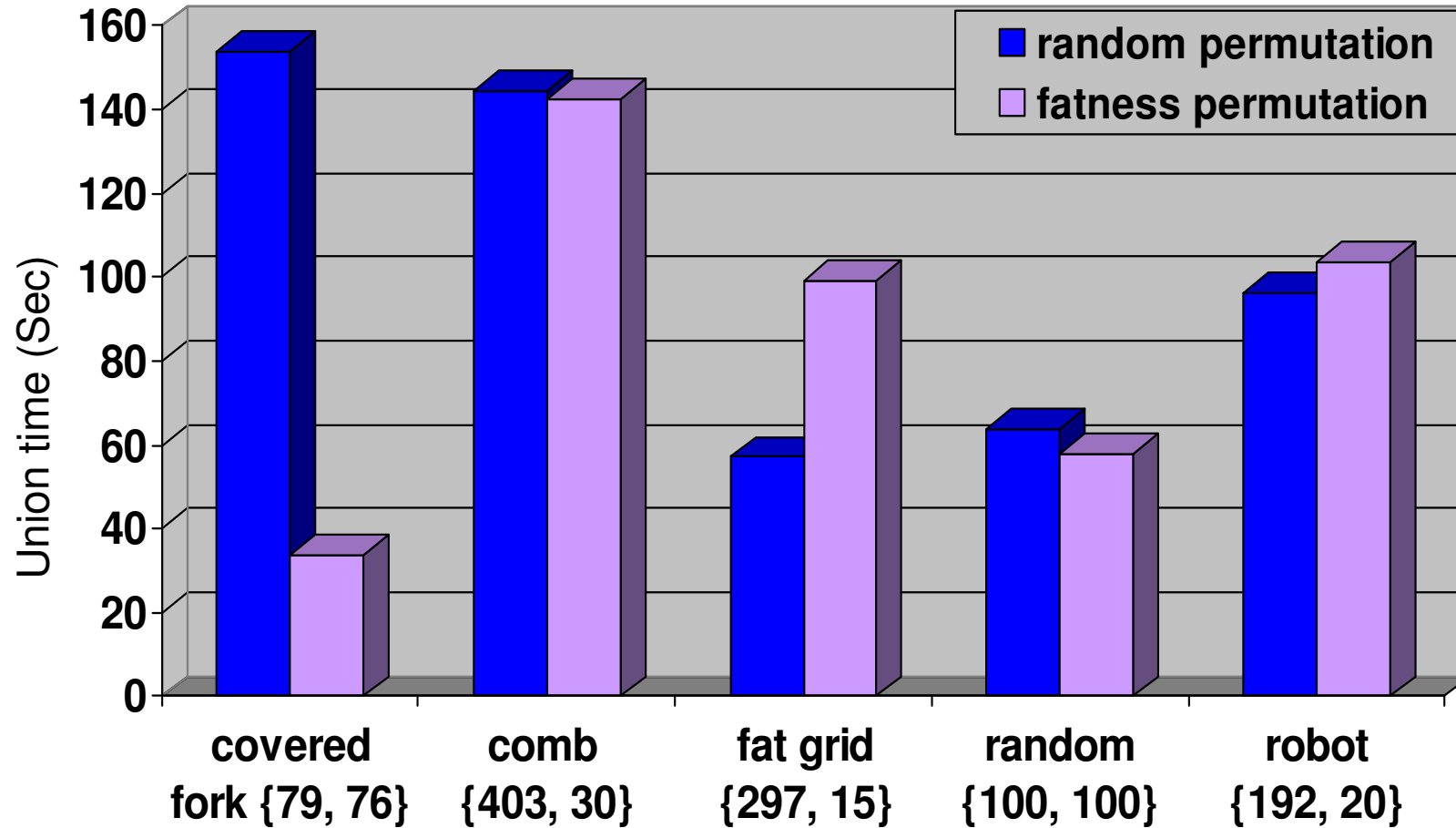
- when the Minkowski sum is relatively complex (larger C_{PQ}) then the *arrangement* algorithm performs better
- when C_{PQ} is small we can save time by removing the non-relevant edges as we do in the *incremental* union algorithm
- the performance of the *divide and conquer* algorithm is mostly between the other two algorithms

Order of insertion



idea: use **fatness** ordering to get output sensitivity effect

Order of insertion - results



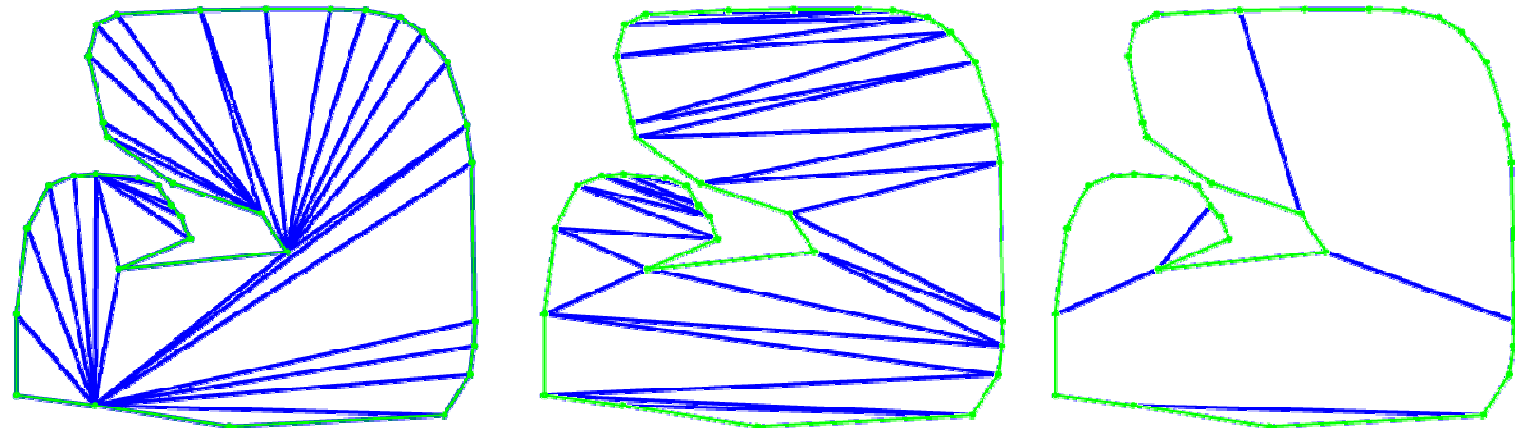
The practice of the decomposition framework

- Step 1 Decompose P and Q into convex subpolygons P_1, \dots, P_s and Q_1, \dots, Q_t
- Step 2 Compute $P_i \oplus Q_j$ for each pair
- Step 3 Construct the union of those subsums

Steps 1 and 3 are (were) challenging from a practical point of view. The main issues:

- union strategy
- handle degeneracies correctly in the union computation
- **suitable decomposition**

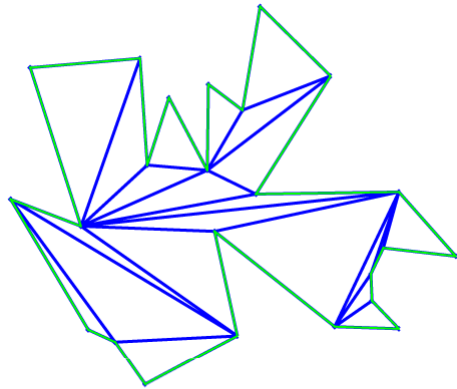
Motivation



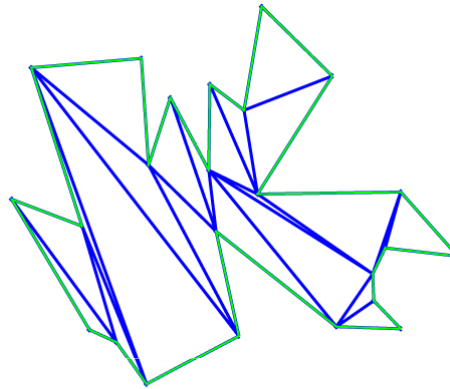
	naive triang.	min $\sum d_i^2$ triang.	min convex
$\sum d_i^2$	754	530	192
# parts	33	33	6
Mink. sum time	2133	1603	120

Time in milli-seconds for computing the Minkowski sum of the polygon with a small convex polygon with 4 vertices

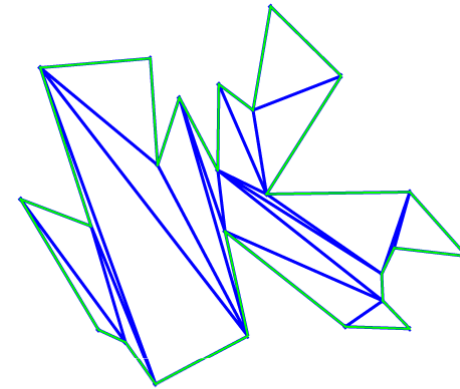
Triangulations



Naive triangulation:
extend a diagonal
from each vertex until
we get a triangulation

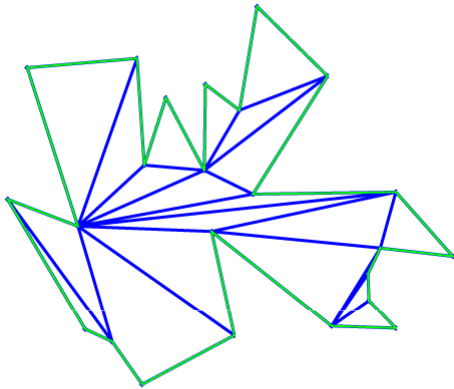


Optimal triangulation:
**minimizing the
maximum degree:**
using dynamic
programming, $O(n^3)$
[KB92]

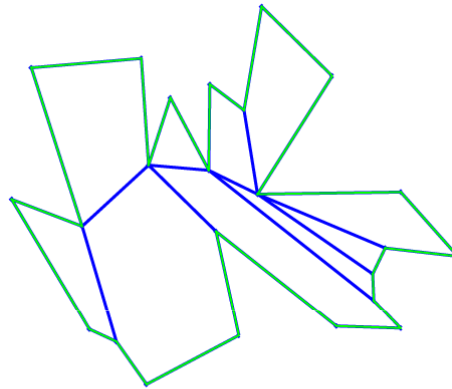


Optimal triangulation:
minimizing $\sum d_i^2$:
a modification of the min-
max-degree triangulation

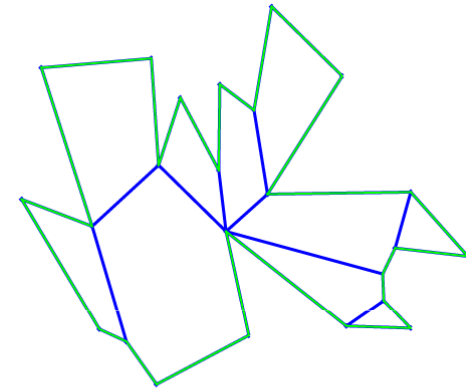
Convex decompositions (no Steiner points)



Greedy convex decomposition:
extend a diagonal from each vertex until we get only convex subpolygons

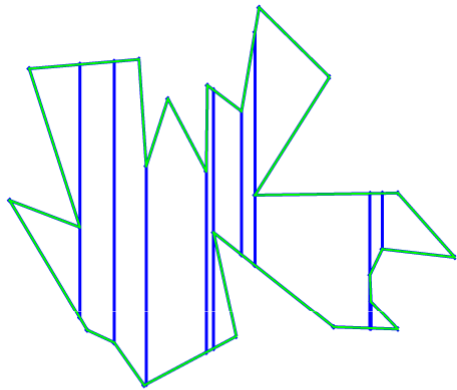


Optimal decomposition: minimum number of convex subpolygons:
using dynamic programming, $O(r^2 n \log n)$ [Keil85]

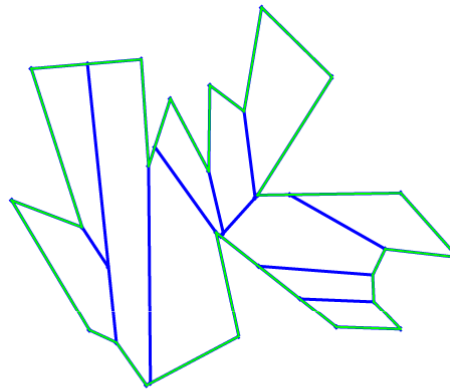


Optimal decomposition: minimum $\sum d_i^2$ convex decomposition:
a modification of Keil's optimal convex decomposition

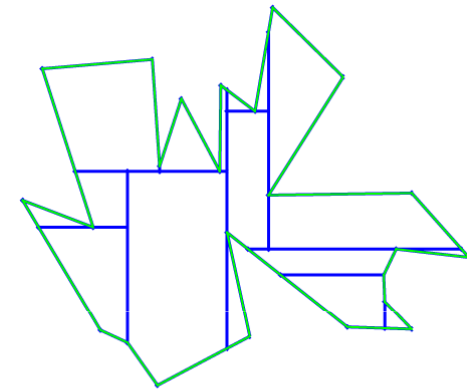
Convex decompositions (allowing Steiner points)



Slab decomposition:
extend upward and downward a vertical segment from each reflex vertex

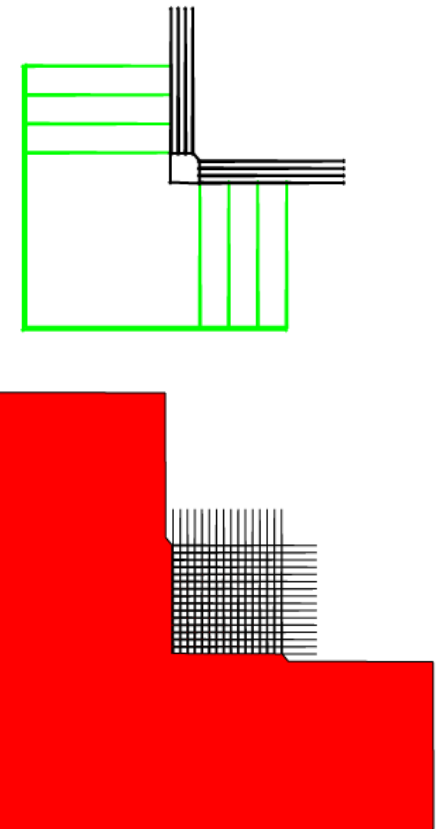
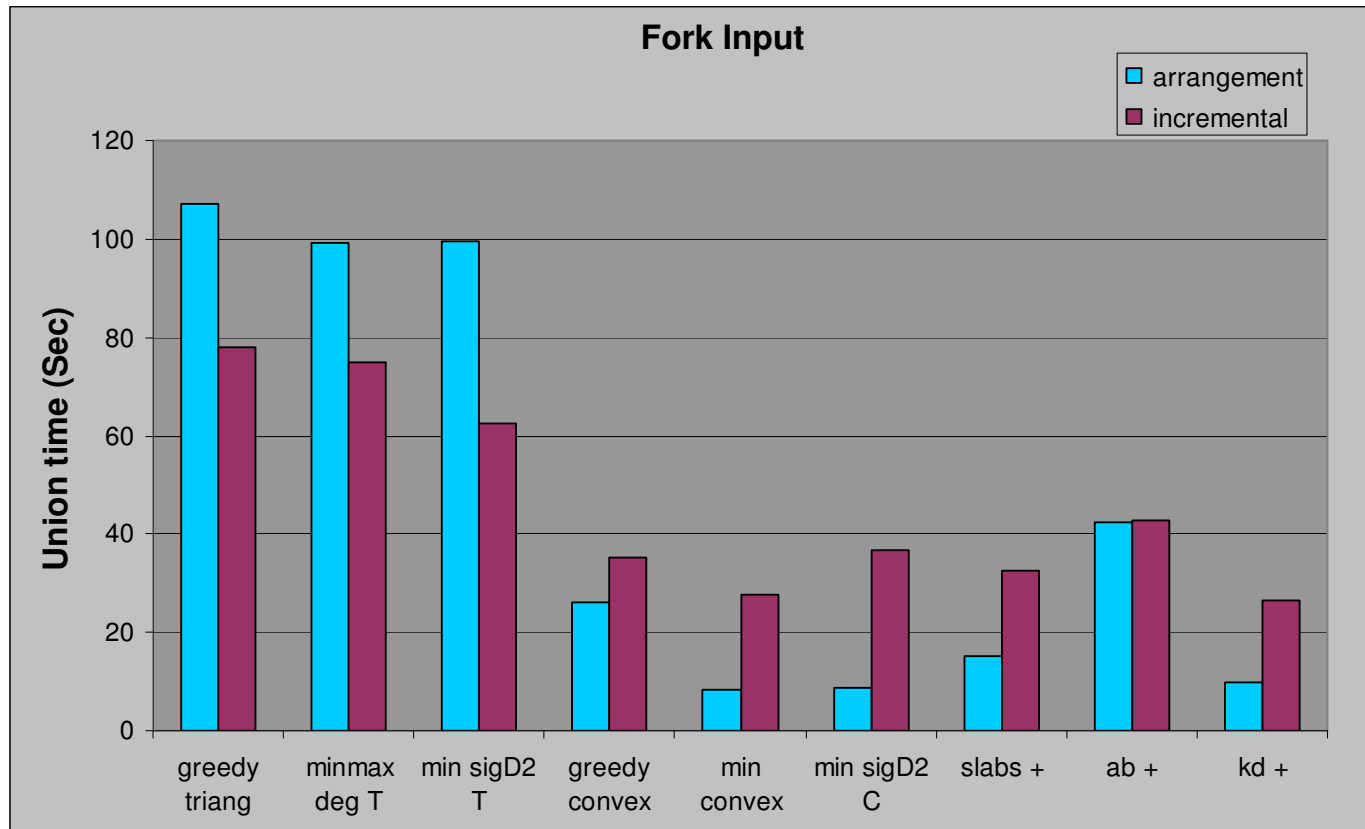


Angle "bisector" decomposition:
extend an angle bisector from each reflex vertex. Gives a 2-approximation for the min-convex decomposition (w/ Steiner points) [CD85]

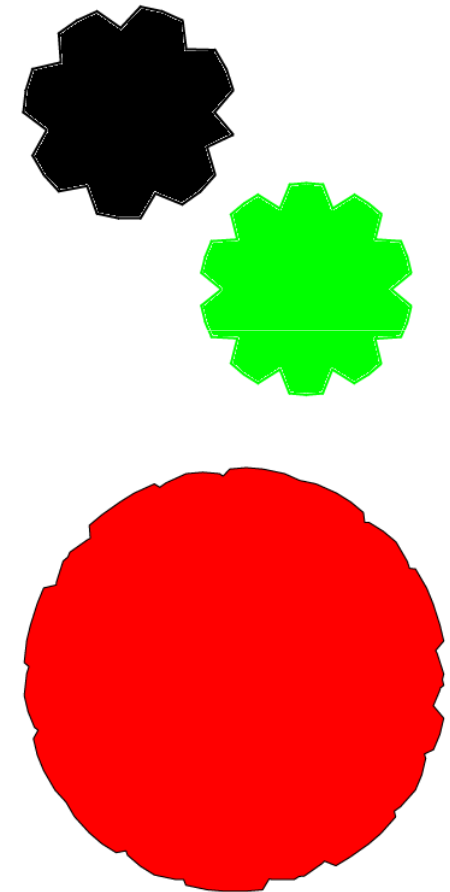
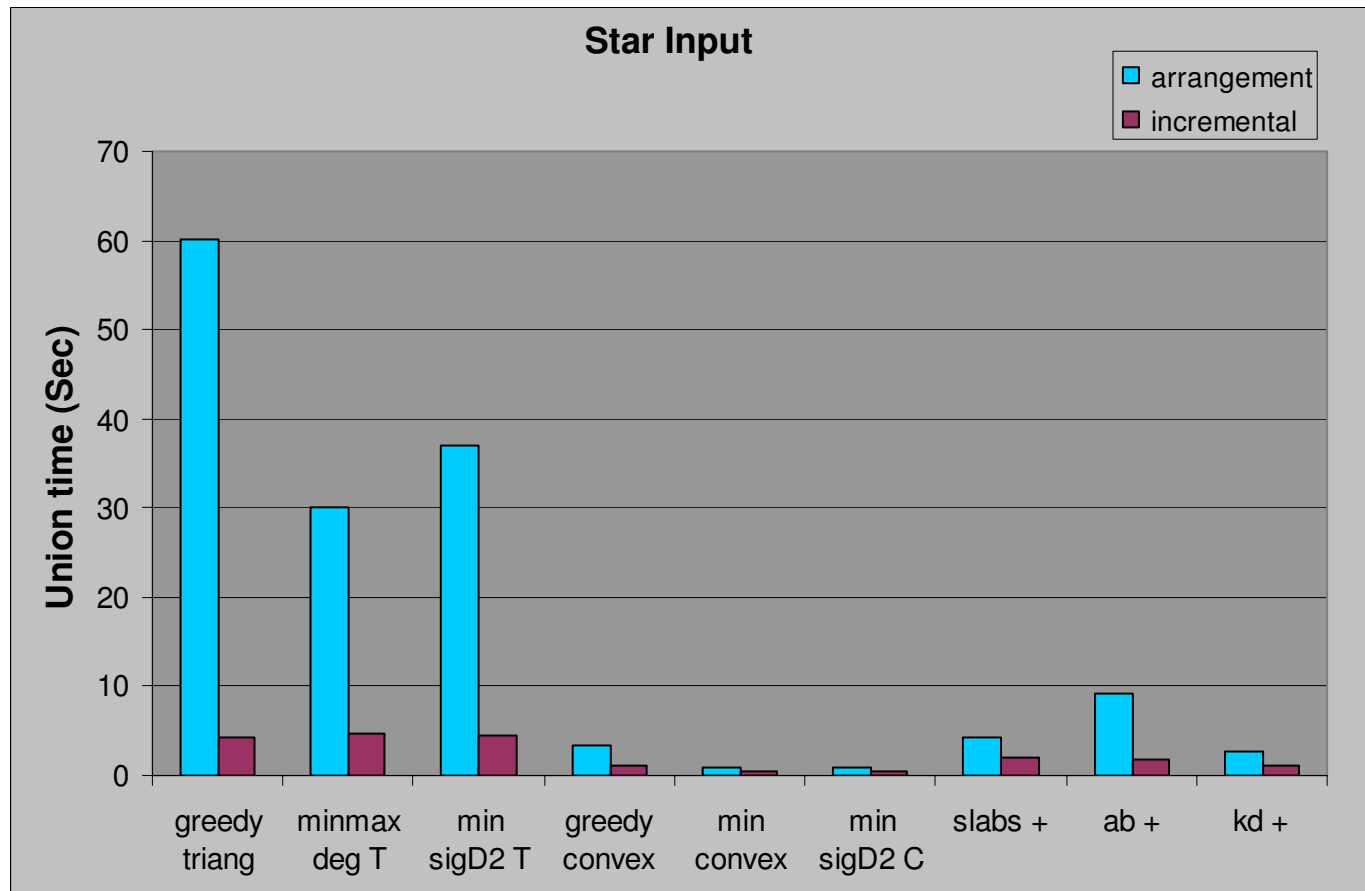


KD decomposition:
extend vertical or horizontal segments from reflex vertices following the KD-tree construction schema

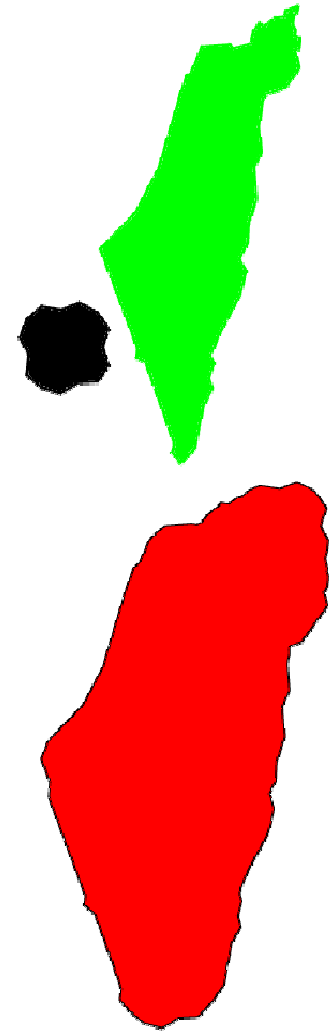
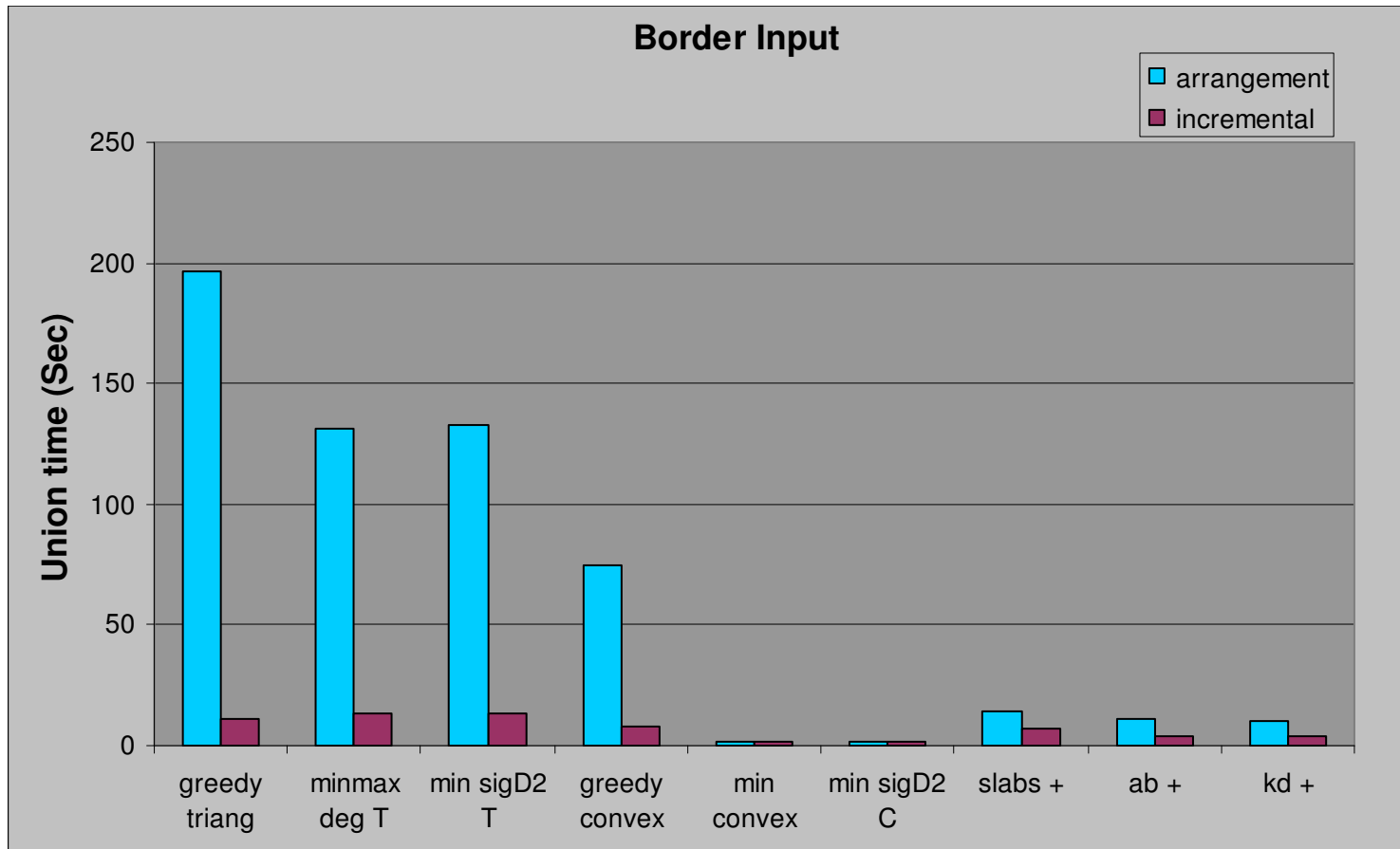
Results: fork input



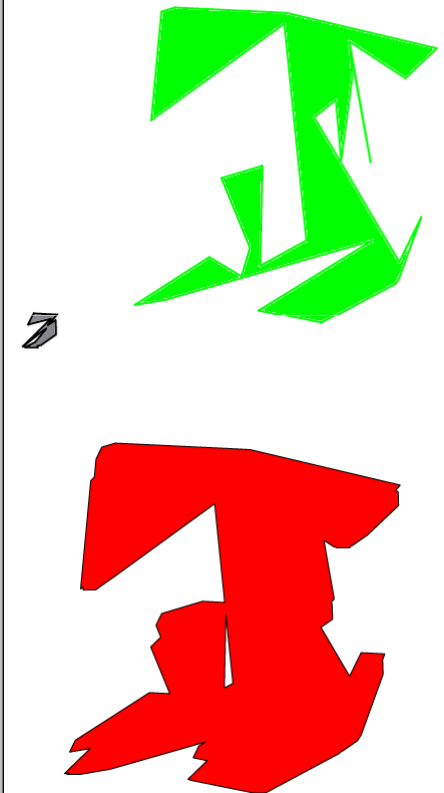
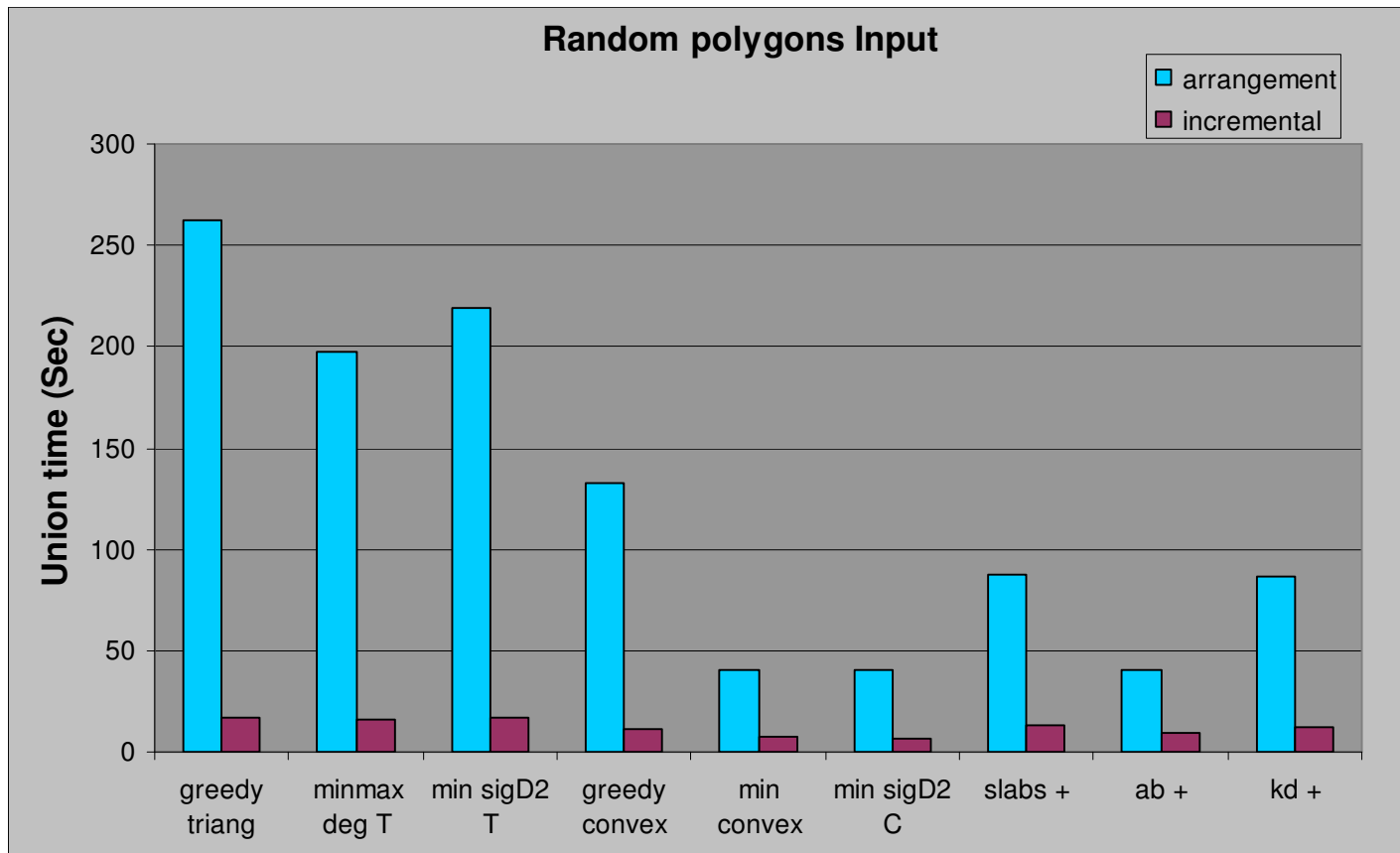
Results: star input



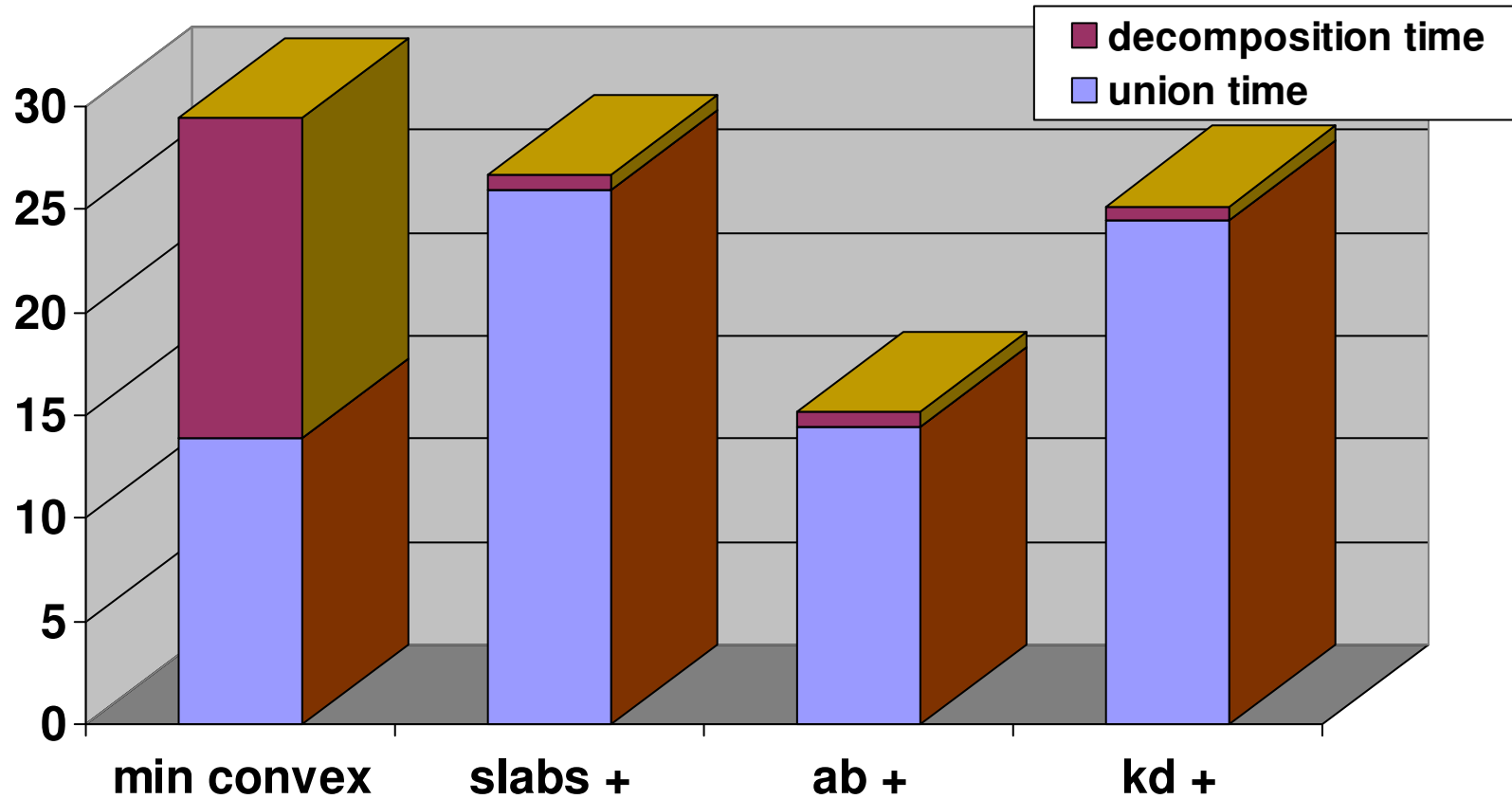
Results: countries borders input



Results: random looking polygons input



Min-convex costs

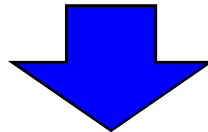


Times for computing the Minkowski sum of two star shaped polygons with 100 vertices each

First round conclusions

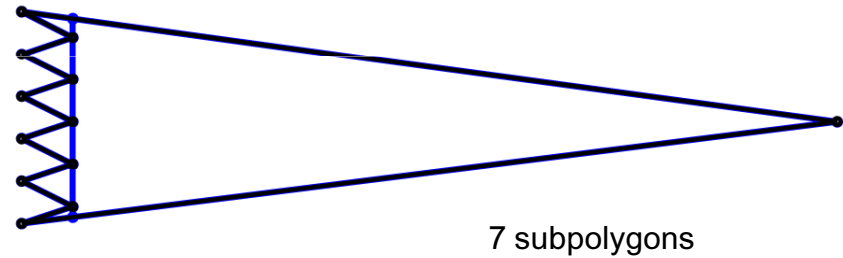
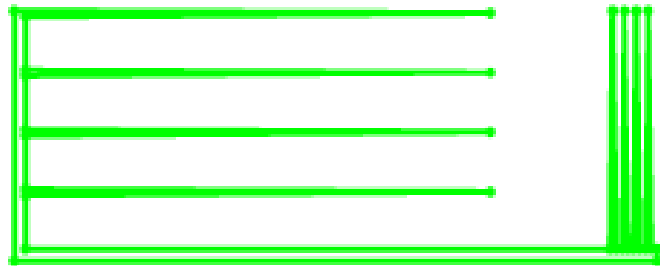
- triangulations give poor results
- min-convex is almost always best
- computing the optimal decompositions can take more time than computing the union

- continue with: min-convex, slab, AB, KD

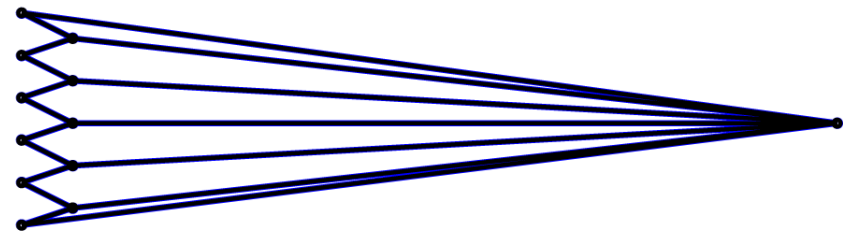


Nonoptimality of min-convex

- Minimizing the number of convex subpolygons is not always the best strategy:

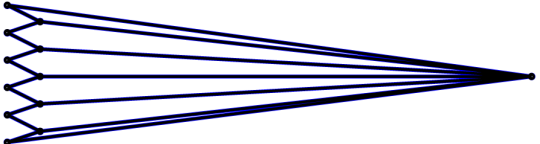
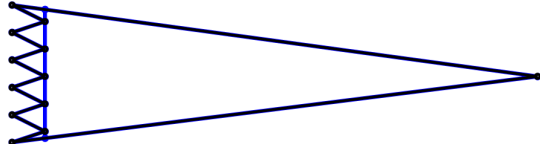


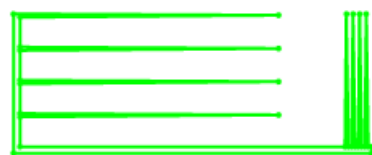
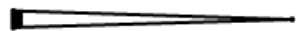
7 subpolygons



6 subpolygons

Nonoptimality of min-convex (contd.)

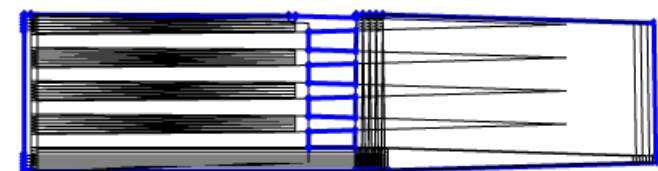
		
No. of vertices in the underlying arr.	23448	9379
Minkowski sum running time (sec)	71.7	25.6



knife input



Minkowski sum



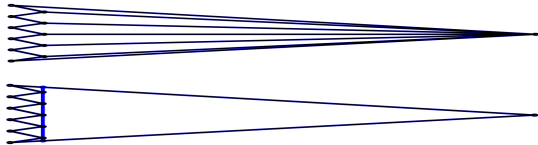
underlying arr.

Mixed decomposition

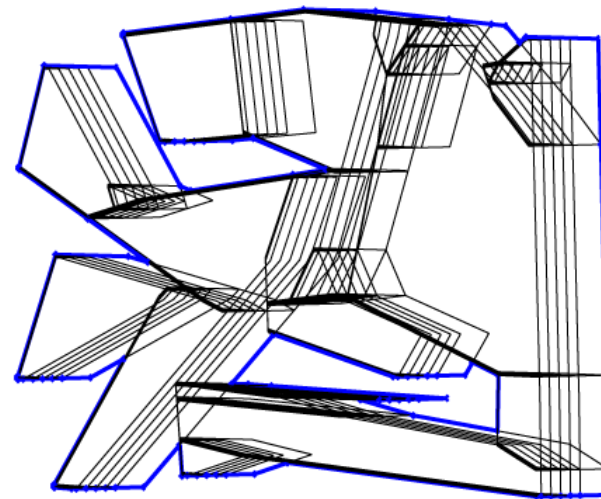
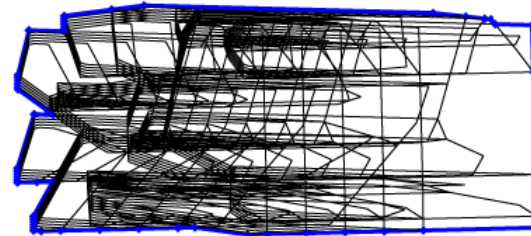
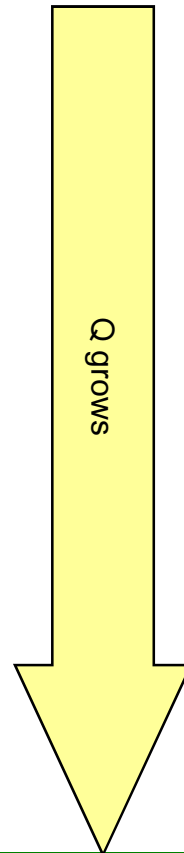
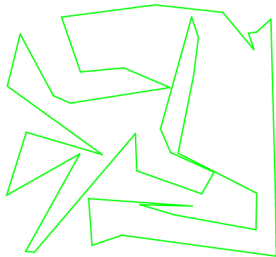
- decomposition techniques that handle P and Q separately might not be sufficient
- according to the previous results, we wish to consider the overall **length** of the decomposition

Decomposition length effect: an example

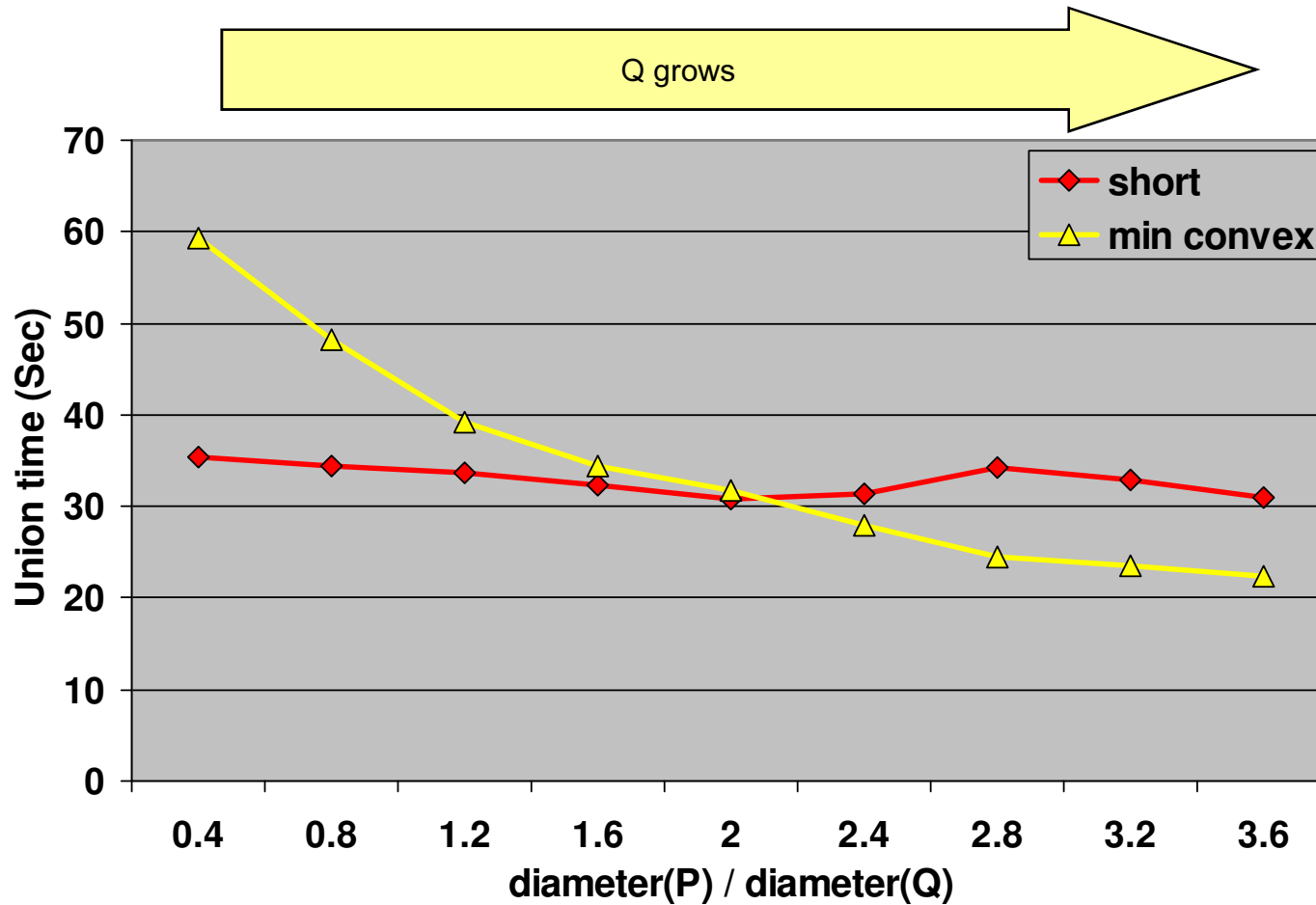
P - fixed size, two types of decompositions



Q - fixed decomposition, scaled size



Decomposition length effect: results



time for computing the Minkowski sum of a knife polygon P (using two types of decompositions) with a random polygon Q that is scaled differently

Mixed objective function - motivation

Time of the arrangement union algorithm:

$$O(I + k \log k)$$



I is the number of intersections among edges of R ; it is harder to optimize I

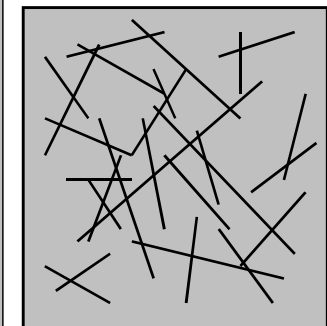
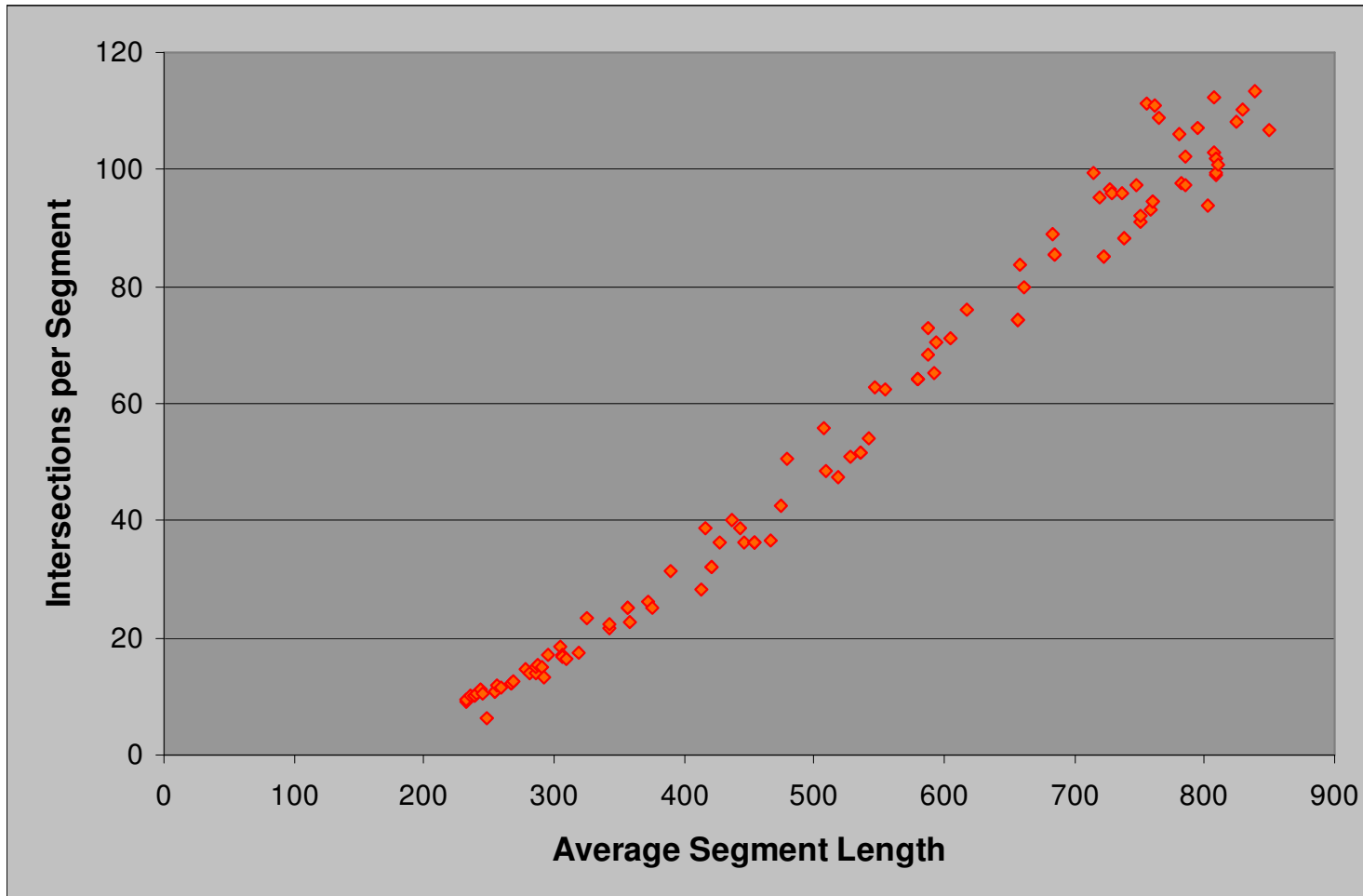


k is the number of edges of R ; we get smaller k for decompositions with lower number of subpolygons.

Smaller number of intersections of segments

- we want each edge of R to intersect as few polygons of R as possible
- $\mu(L(R_{ij}))$ - the standard rigid-motion invariant measure of the set of lines intersecting R_{ij}
- $\mu(L(R_{ij}))$ is the perimeter of R_{ij}

Length vs. number of intersections



The mixed function

$$k_Q(2\Delta_P + \Pi_P) + k_P(2\Delta_Q + \Pi_Q)$$

k_P - number of subpolygons in the convex decomposition of P

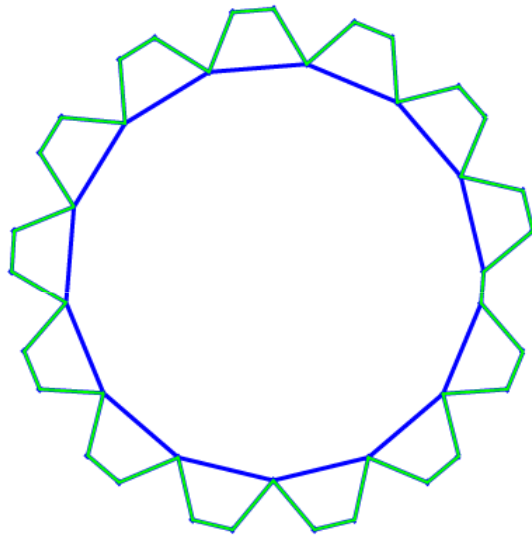
Δ_P - total length of diagonal in the decomposition of P

Π_P - the perimeter of P

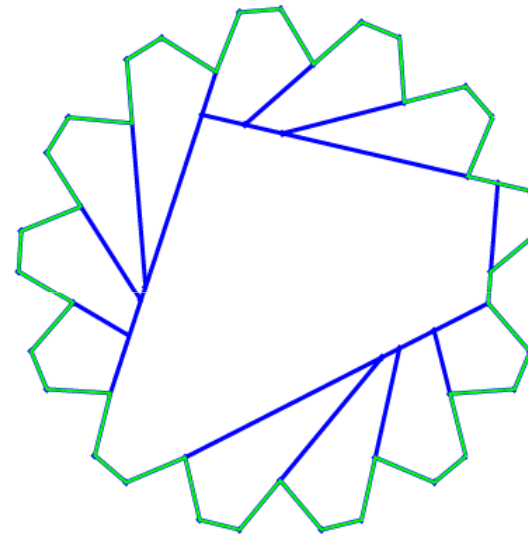
The function measures the overall length of the edges of R .

An $O(n^2r_P^4 + m^2r_Q^4)$ -time decomposition algorithm that minimizes this function (based on [Keil85])

Improved AB algorithms

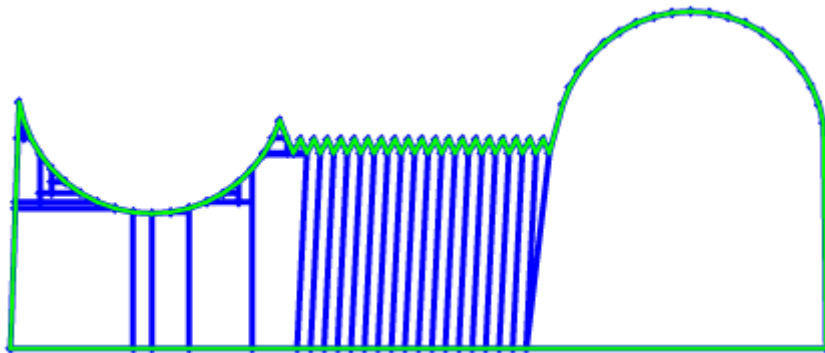


Minimal length AB decomposition: from each reflex vertex, extend the shorter: an angle bisector or short diagonal

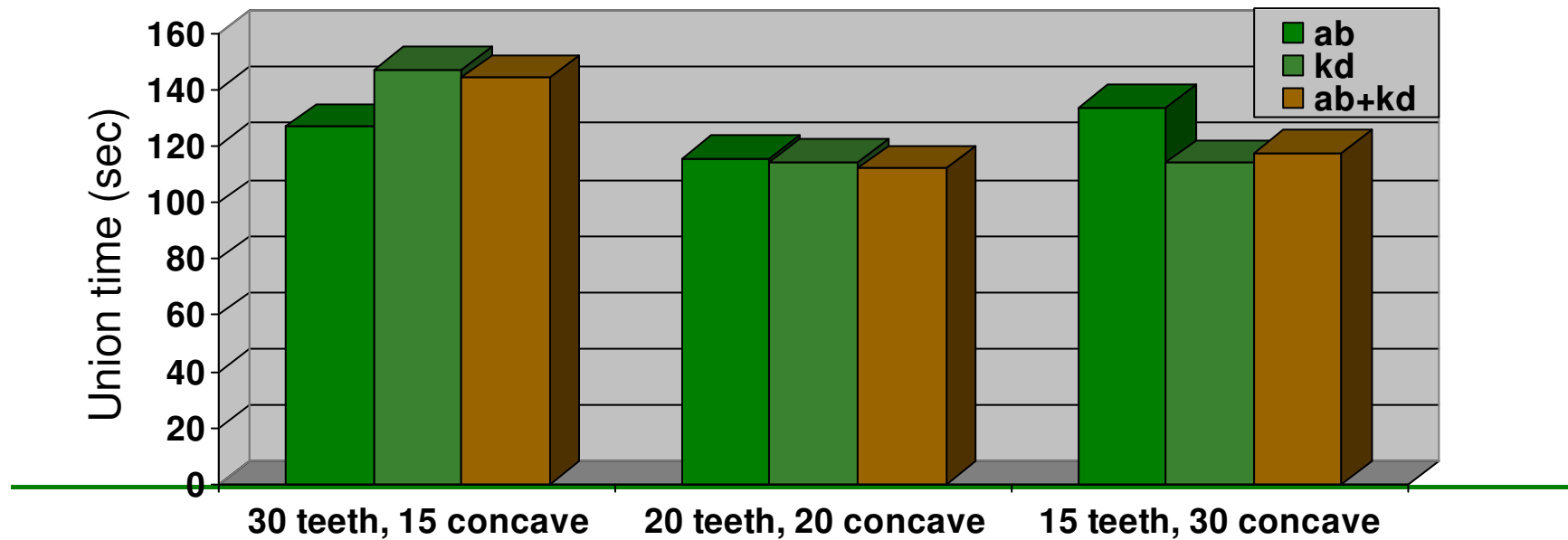


Improved/Reflex AB decomposition: look harder for 2-reflex eliminators

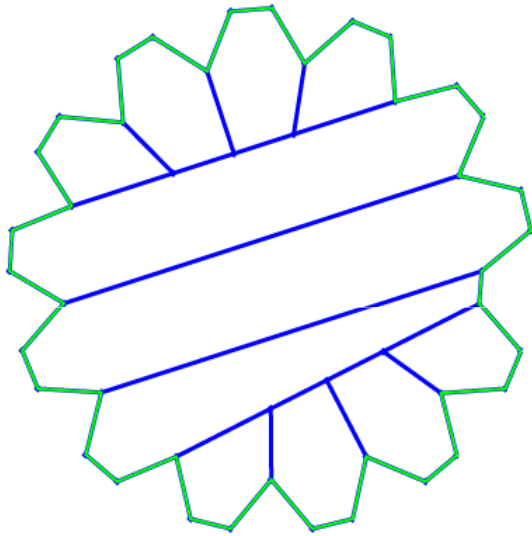
Improved AB & KD algorithms (contd.)



Composite AB+KD decomposition: use KD decomposition for concave chains and AB for the rest of the reflex vertices

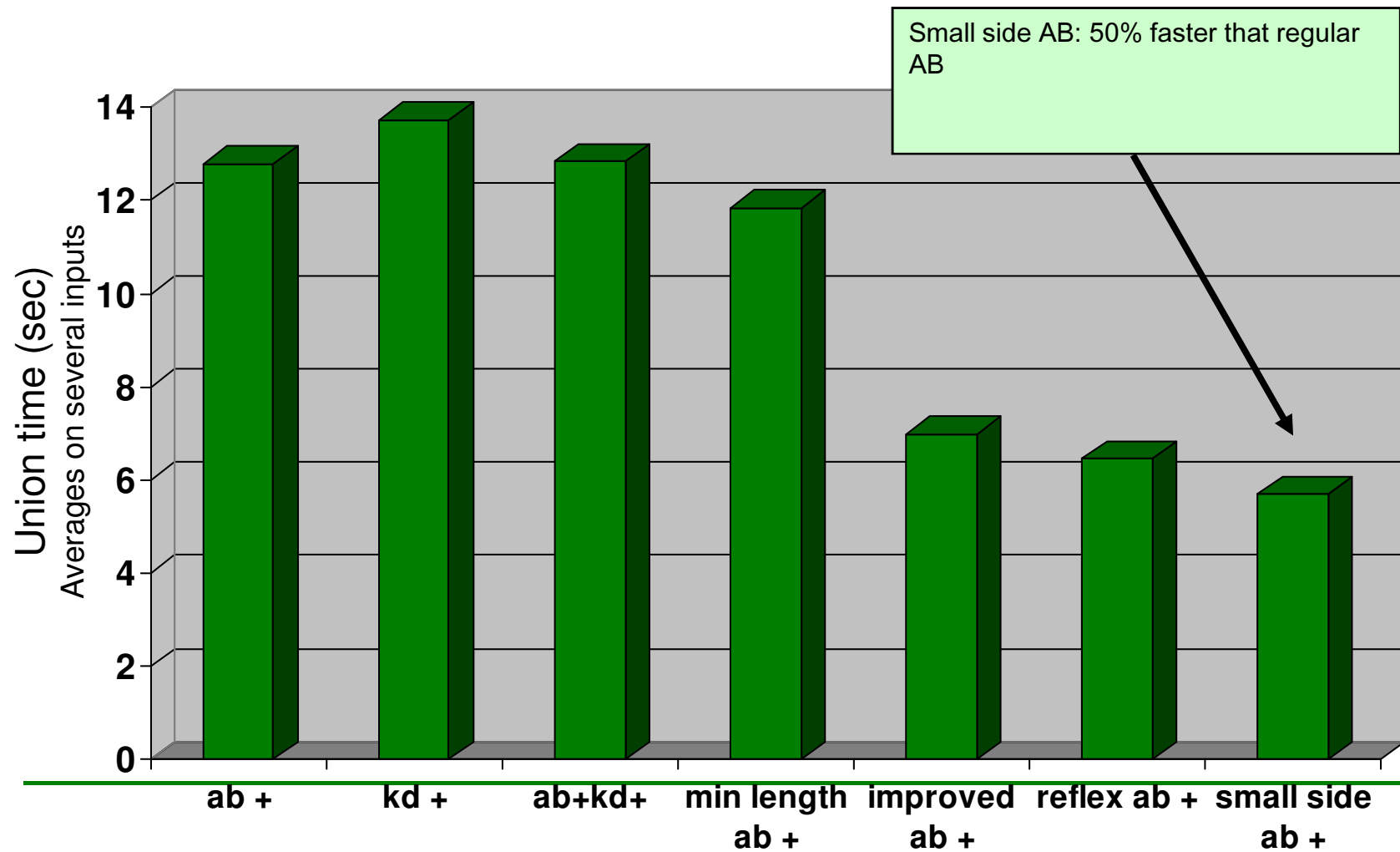


Small side AB



Small side AB decomposition: we look for 2-reflex eliminators that “block” the minimal number of reflex vertices

Decomposition improvements: results



Alternative approach to computing Minkowski sums: the convolution method

all existing methods traverse an arrangement in the final stage, deciding the features that participate in the sum; the difference is in how the arrangement is constructed

- take I: computing all critical curves
- take II: computing convolution cycles

Offset polygons

popular form of Minkowski sums: the sum of a polygon and a disc

- offset polygon for a convex polygon is easy to compute
- decomposition approach applies with higher-degree algebra

References

- Most of the presentation is based on
 - [Agarwal-Flato-H '02]
Polygon decomposition for efficient construction of Minkowski sums, CGTA
 - Chapter 13 of the book “Computational Geometry” by de Berg et al
 - The “convolution” approach
 - [Guibas-Ramshaw-Stolfi '83]
A kinetic framework for computational geometry, FOCS
 - [Guibas-Seidel '85]
Computing convolutions by reciprocal search, DCG
 - [Wein '07]
Exact and approximate construction of offset polygons, CAD
 - CGTA = Computational Geometry, Theory & Applications
 - CAD = Computer-Aided-Design Journal
 - DCG = Discrete and Computational Geometry
 - FOCS = IEEE Symp. Foundations of Computer Science
-



THE END