Covering Polygons with Rectangles

Roland Glück
roland.glueck@dlr.de
Deutsches Zentrum für Luft- und Raumfahrt
Bern, 22nd April 2017

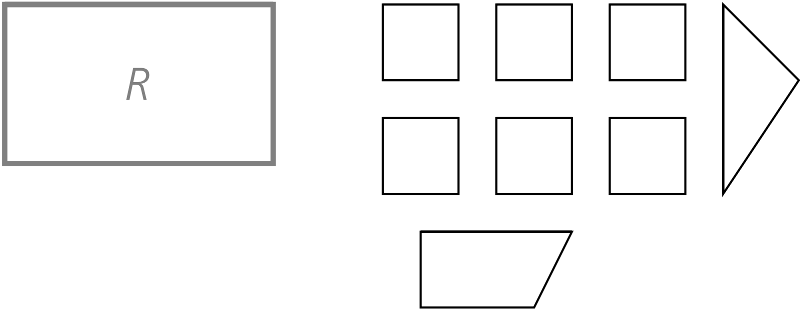
Motivation: Cutting Center

Basic Definitions

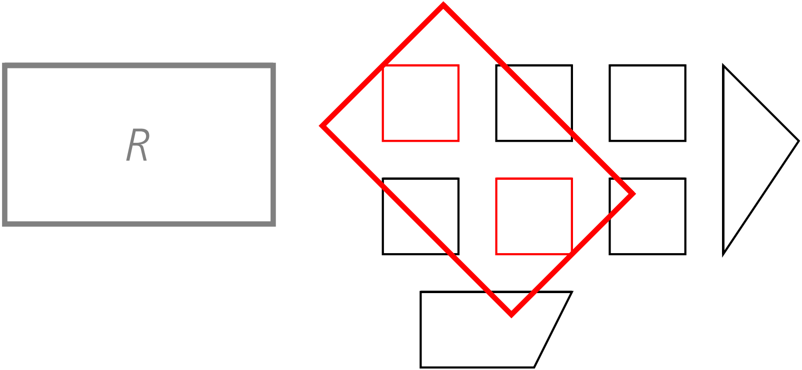
Definition

- *Nesting* $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ is a set of n simple polygons.
- *Cover* $\mathbf{C} = \{R_1, R_2, \ldots, R_m\}$ of a nesting $\mathbf{P}$ by a gripper R is a set of rectangles such that
  - every R_j is a copy of R, and
  - every polygon $P_i \in \mathbf{P}$ is contained completely in at least one $R_j \in \mathbf{C}$.
- Problem: find an *optimal cover*, i.e. a cover with minimum cardinality
- Copies of R arise from R by
  - translation along the x-axis (easy to solve)
  - arbitrary translation (NP-hard)
  - arbitrary translation and rotation (NP-hard, considered here)

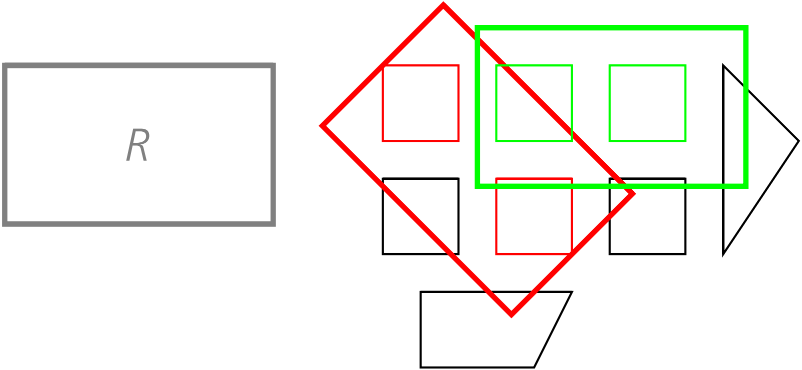
Example Cover



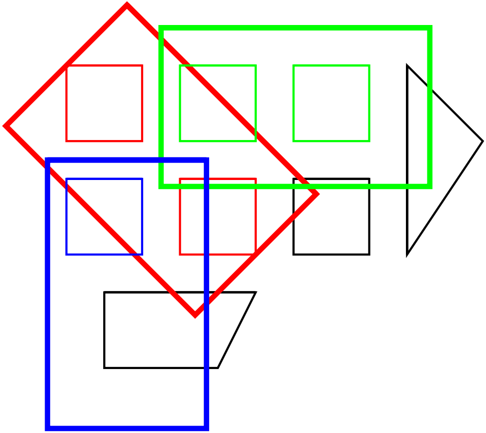
Example Cover



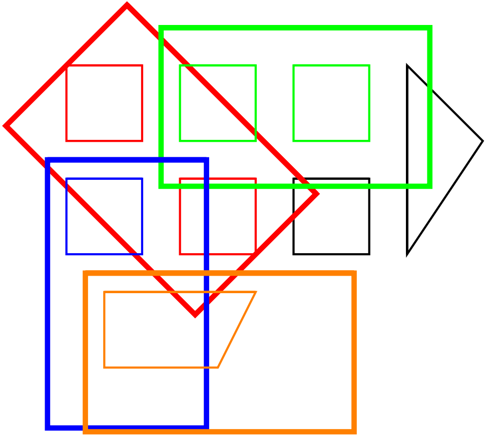
Example Cover



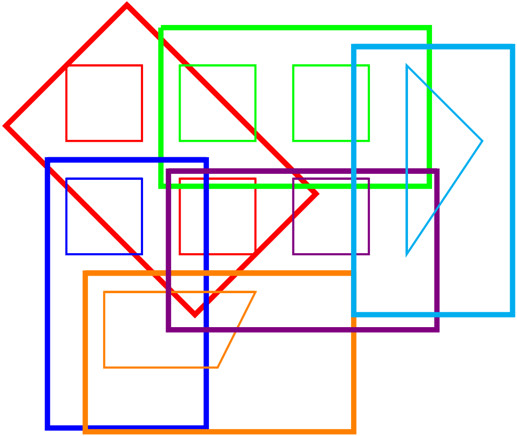
Example Cover



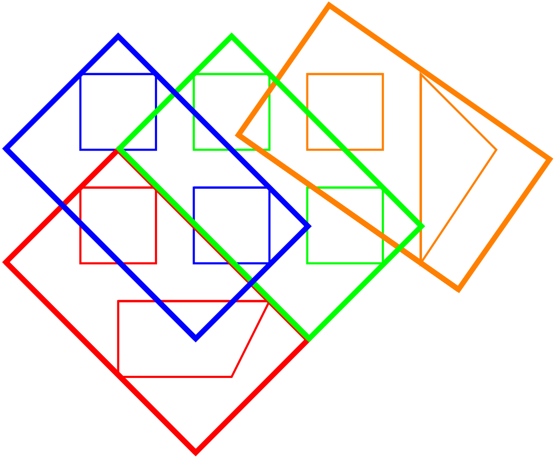
Example Cover



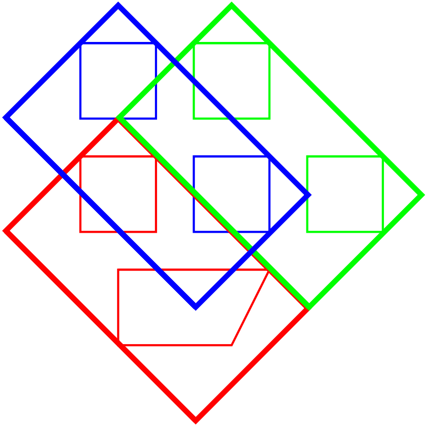
Example Cover



Optimal Cover



Decomposition of an Optimal Cover



Decomposition Property

- Notation: given a nesting **P** and a rectangle R, we denote by $\mathrm{cov}(R, \mathbf{P})$ the set of all polygons of **P** which are contained completely in R.

Decomposition Property

- Notation: given a nesting $\mathbf{P}$ and a rectangle R, we denote by $\mathrm{cov}(R, \mathbf{P})$ the set of all polygons of $\mathbf{P}$ which are contained completely in R.

Lemma

Let $\mathbf{C} = \{R_1, R_2, \ldots, R_m\}$ be an optimal cover of $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ and choose an arbitrary $R_j \in \mathbf{C}$. Then $\mathbf{C} \backslash \{R\}$ is an optimal cover of $\mathbf{P} \backslash \mathrm{cov}(R, \mathbf{P})$.

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...
- ... proceed recursively with the remaining nesting ...

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...
- ... proceed recursively with the remaining nesting ...
- ... till we obtain the empty nesting.

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...
- ... proceed recursively with the remaining nesting ...
- ... till we obtain the empty nesting.

Work with a *set* of **candidate rectangles** containing at least one rectangle from an optimal cover.

Basic Idea

Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...
- ... proceed recursively with the remaining nesting ...
- ... till we obtain the empty nesting.

Work with a *set* of **candidate rectangles** containing at least one rectangle from an optimal cover.

- **Idea:** Every polygon has to be contained completely in at least one rectangle.

Basic Idea

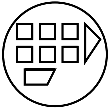
Assume we could compute for every nesting one rectangle of an optimal cover. Then ...

- ... we could remove the rectangle and all polygons contained completely in it, ...
- ... proceed recursively with the remaining nesting ...
- ... till we obtain the empty nesting.

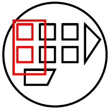
Work with a *set* of **candidate rectangles** containing at least one rectangle from an optimal cover.

- **Idea:** Every polygon has to be contained completely in at least one rectangle.
- Choose pivot polygon P_{pi} and compute $\subseteq$-maximal subsets of $\mathbf{P}$ containing P_{pi} which are contained in a copy of R.

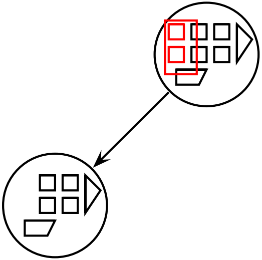
Search Graph



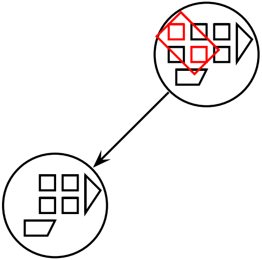
Search Graph



Search Graph

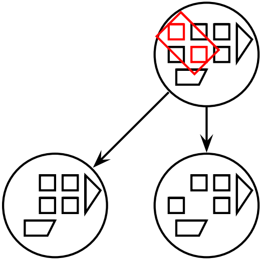


Search Graph

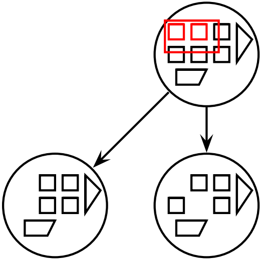


DLR

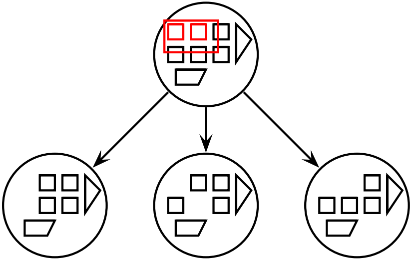
Search Graph



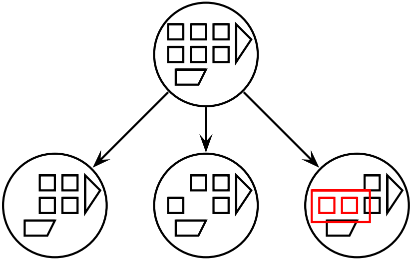
Search Graph



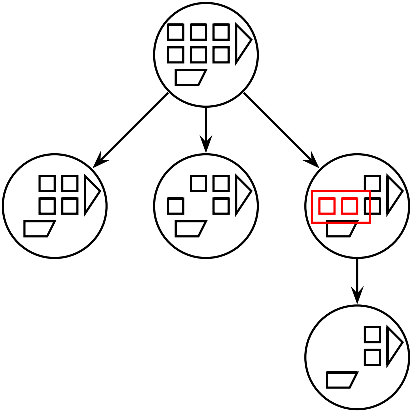
Search Graph



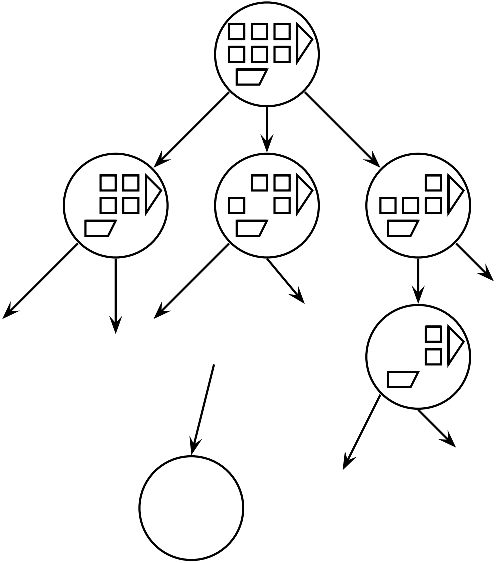
Search Graph



Search Graph



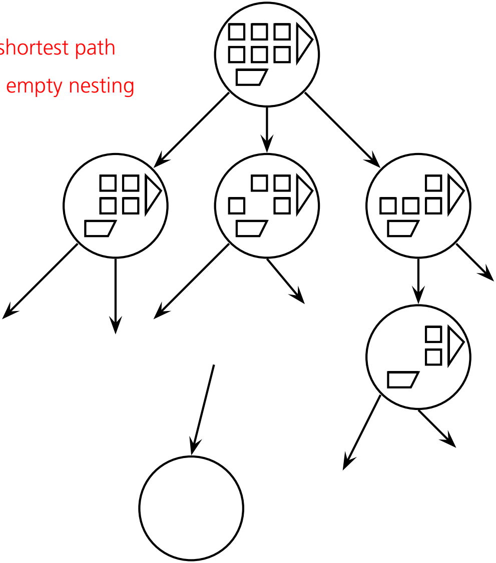
Search Graph



Search Graph

Goal: find shortest path

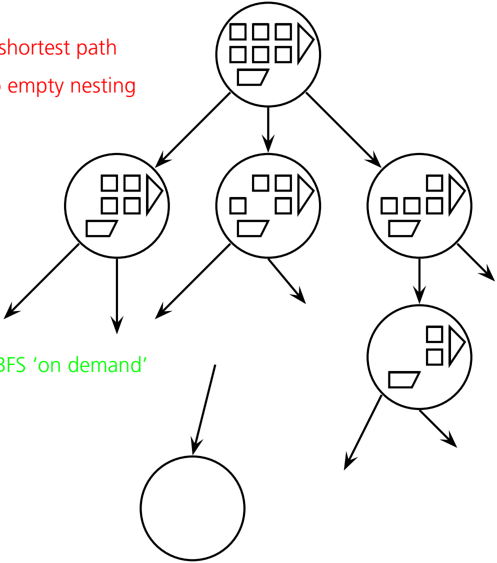
from root to empty nesting



Search Graph

Goal: find shortest path

from root to empty nesting



Solution: BFS 'on demand'

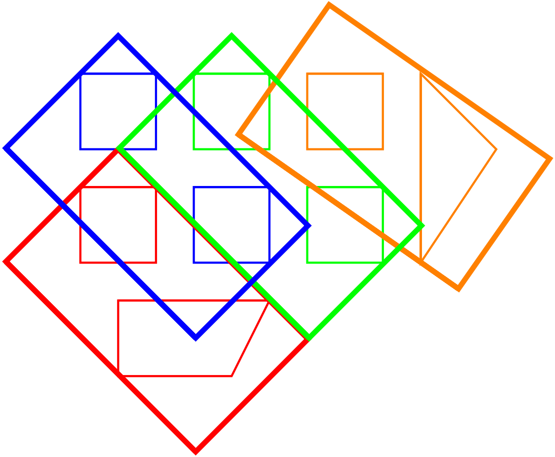
Computation of Candidate Rectangles

How to compute the candidate rectangles?

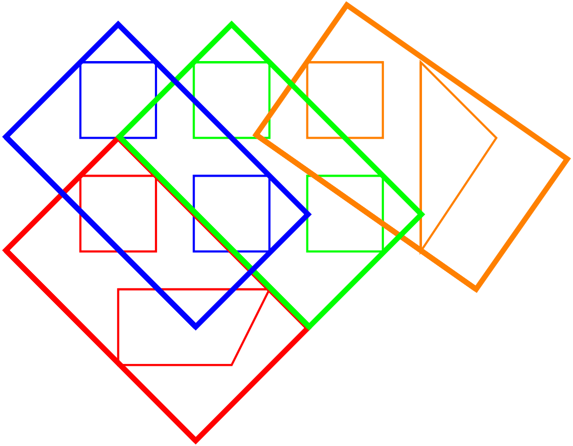
Two approaches:

- vertex-oriented approach
- polygon-oriented approach

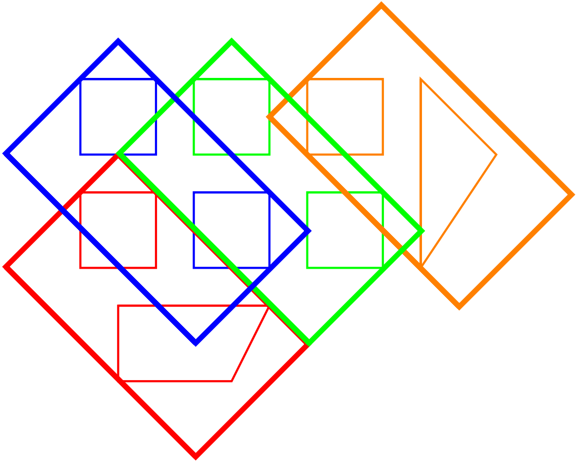
Vertex-Oriented Approach - Alignment of a Rectangle



Vertex-Oriented Approach - Alignment of a Rectangle



Vertex-Oriented Approach - Alignment of a Rectangle



Vertex-Oriented Approach

Observation: Every rectangle R of an optimal cover can be replaced by a rectangle where three vertices lie on at least two different adjacent sides of R, without destroying optimality (degenerated cases possible but easy to handle).

Vertex-Oriented Approach

Observation: Every rectangle R of an optimal cover can be replaced by a rectangle where three vertices lie on at least two different adjacent sides of R, without destroying optimality (degenerated cases possible but easy to handle).

Hence:

- loop over all triples (v_1, v_2, v_3) of vertices of a nesting

Vertex-Oriented Approach

Observation: Every rectangle R of an optimal cover can be replaced by a rectangle where three vertices lie on at least two different adjacent sides of R, without destroying optimality (degenerated cases possible but easy to handle).

Hence:

- loop over all triples (v_1, v_2, v_3) of vertices of a nesting
- align R on (v_1, v_2, v_3) according to above

Vertex-Oriented Approach

Observation: Every rectangle R of an optimal cover can be replaced by a rectangle where three vertices lie on at least two different adjacent sides of R, without destroying optimality (degenerated cases possible but easy to handle).

Hence:

- loop over all triples (v_1, v_2, v_3) of vertices of a nesting
- align R on (v_1, v_2, v_3) according to above
- determine the set of completely contained polygons

Vertex-Oriented Approach

Observation: Every rectangle R of an optimal cover can be replaced by a rectangle where three vertices lie on at least two different adjacent sides of R, without destroying optimality (degenerated cases possible but easy to handle).

Hence:

- loop over all triples (v_1, v_2, v_3) of vertices of a nesting
- align R on (v_1, v_2, v_3) according to above
- determine the set of completely contained polygons
- keep all rectangles containing completely an $\subseteq$-maximal set of polygons including the pivot polygon (modulo cover equivalent rectangles)

Polygon-Oriented Approach

Idea:

- abstract from rectangles and
- compute all $\subseteq$-maximal sets of polygons contained completely in a copy of R, containing P_{pi}
- by backtracking
- missing jigsaw piece:

Polygon-Oriented Approach

Idea:

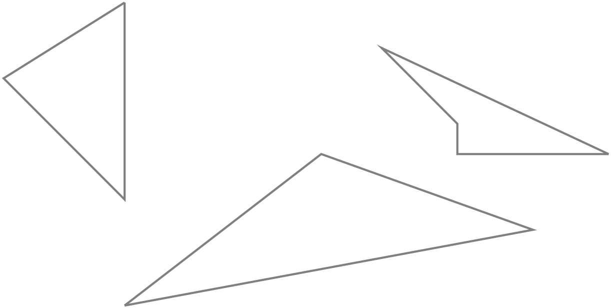
- abstract from rectangles and
- compute all $\subseteq$-maximal sets of polygons contained completely in a copy of R, containing P_{pi}
- by backtracking
- missing jigsaw piece:
  - given a rectangle R and a set of polygons $\{P_1, P_2, \ldots, P_n\}$,
  - determine whether there is a copy of R containing all P_i completely, and,
  - if so, compute one such copy

Polygon-Oriented Approach

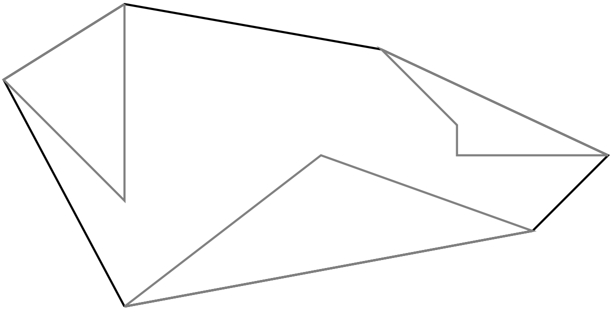
Idea:

- abstract from rectangles and
- compute all $\subseteq$-maximal sets of polygons contained completely in a copy of R, containing P_{pi}
- by backtracking
- missing jigsaw piece:
  - given a rectangle R and a set of polygons $\{P_1, P_2, \ldots, P_n\}$,
  - determine whether there is a copy of R containing all P_i completely, and,
  - if so, compute one such copy
- for a suitable rotational part the translational part is easy to determine
- here: rotate polygon set instead of rectangle for better understanding

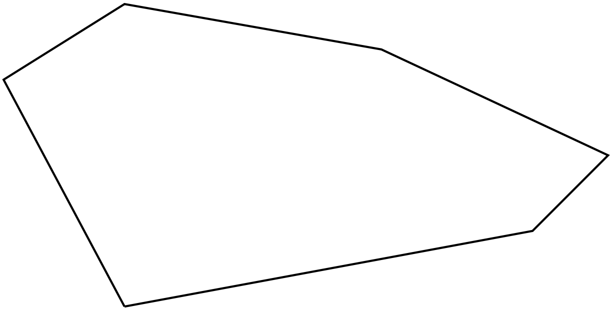
Polygon-Oriented Approach - Convex Hull



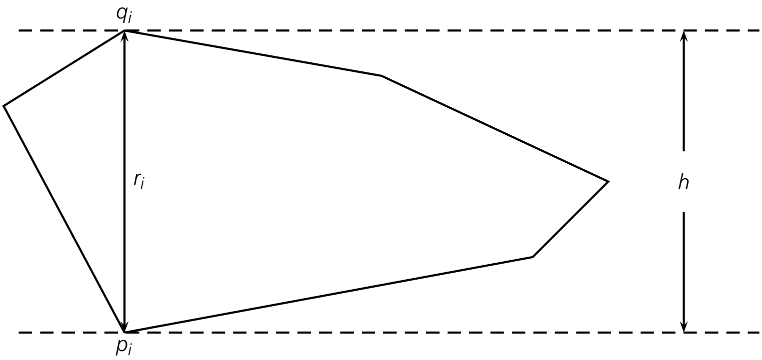
Polygon-Oriented Approach - Convex Hull



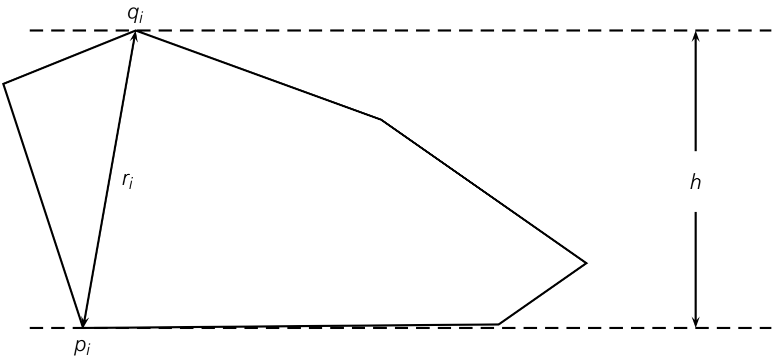
Polygon-Oriented Approach - Convex Hull



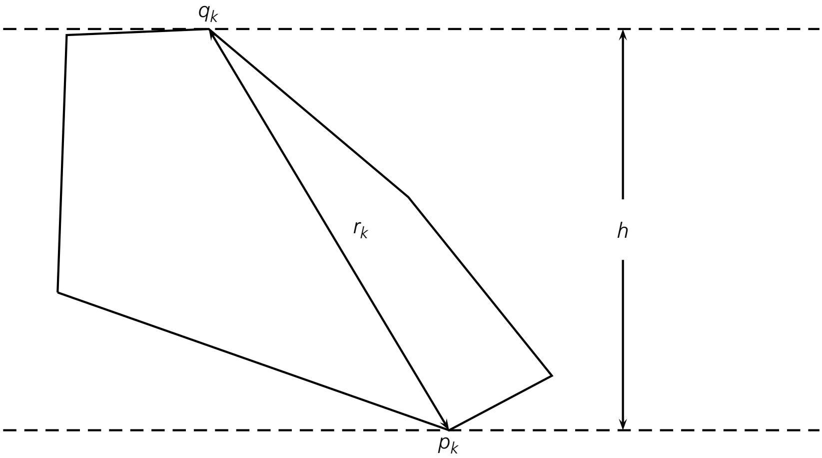
Polygon-Oriented Approach - Determining the Height



Polygon-Oriented Approach - Determining the Height



Polygon-Oriented Approach - Determining the Height



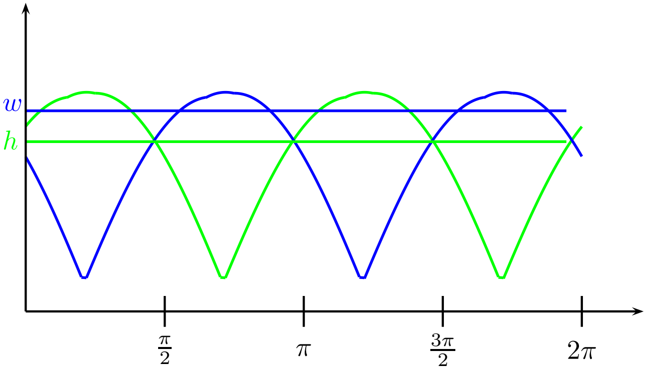
Polygon-Oriented Approach - Intersecting Height and Width Function

- height representable as piecewise defined sine function
- width shifted by $\frac{\pi}{2}$
- determine all intervals where height of the rotated polygon is less-equal to gripper height
- proceed analogously for width
- intersect admissible height- and width intervals

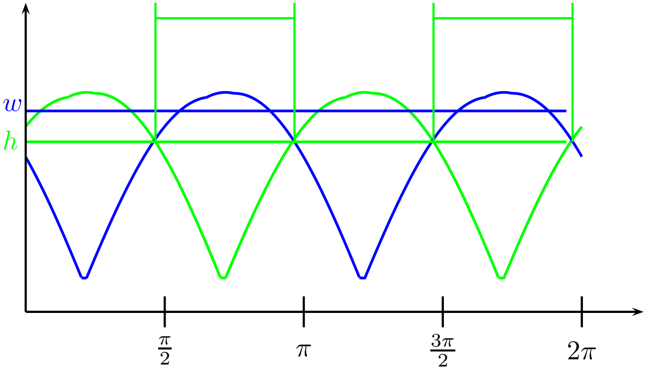
Polygon-Oriented Approach - Intersecting Height and Width Function

- height representable as piecewise defined sine function
- width shifted by $\frac{\pi}{2}$
- determine all intervals where height of the rotated polygon is less-equal to gripper height
- proceed analogously for width
- intersect admissible height- and width intervals
- interval construction in linear time by means of rotating calipers
- intersection in linear time by means of sweep line
- linear overall running time

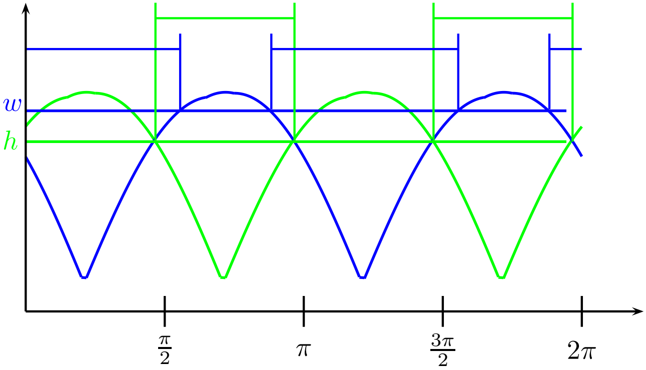
Polygon-Oriented Approach - Graphic Representation



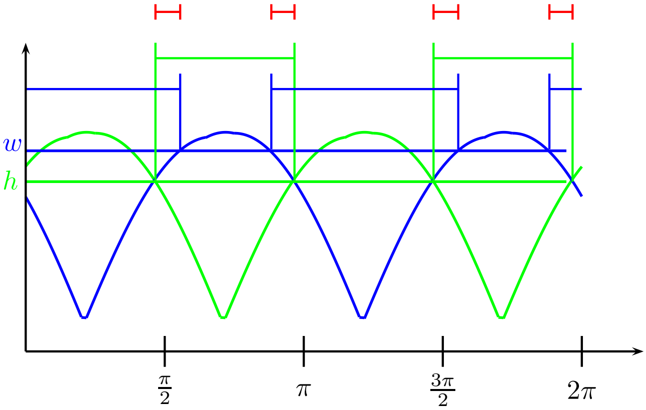
Polygon-Oriented Approach - Height Intervals



Polygon-Oriented Approach - Height and Width Intervals



Polygon-Oriented Approach - Suitable Intervals



Running Time Estimations

Notation: $\|\mathbf{P}\|$ denotes overall number of vertices (clearly, $|\mathbf{P}|$ denotes number of polygons)

Running Time Estimations

Notation: $||\mathbf{P}||$ denotes overall number of vertices (clearly, $|\mathbf{P}|$ denotes number of polygons)

- vertex-oriented approach loops roughly over all triples of vertices
- depends only loosely on number of polygons
- running time $\sim O(||\mathbf{P}||^3)$
- well-suited for instances with high number of polygons and low number of vertices

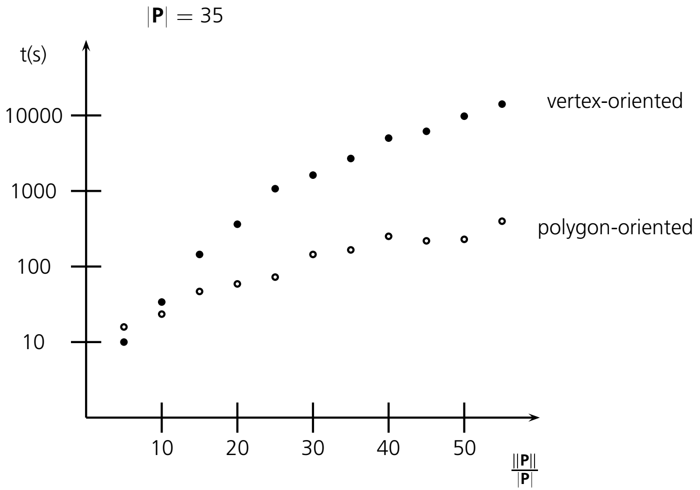
Running Time Estimations

Notation: $||\mathbf{P}||$ denotes overall number of vertices (clearly, $|\mathbf{P}|$ denotes number of polygons)

- vertex-oriented approach loops roughly over all triples of vertices
- depends only loosely on number of polygons
- running time $\sim O(||\mathbf{P}||^3)$
- well-suited for instances with high number of polygons and low number of vertices

- polygon-oriented approach needs to examine $2^{|\mathbf{P}|}$ combinations of polygons
- but running time linear in $||\mathbf{P}||$
- well-suited for instances with high number of vertices and low number of polygons

Experimental Results



Experimental Results

$\lvert \mathbf{P} \rvert$	$\frac{\lVert \mathbf{P} \rVert}{\lvert \mathbf{P} \rvert}$	polygon-oriented		vertex-oriented			
		μ	c_V	μ	c_V	$\left(\frac{\lVert \mathbf{P}_n \rVert}{\lVert \mathbf{P}_{n-1} \rVert}\right)^3$	$\frac{\mu_n}{\mu_{n-1}}$
25	5	1.5	0.28	1.3	0.21	-	-
25	10	1.3	0.34	9.0	0.2	8	6.9
25	15	1.1	0.28	32	0.15	3.4	3.6
25	20	1.8	0.44	78	0.21	2.4	2.4
25	25	2.0	0.49	160	0.16	2.0	2.1
25	30	1.3	0.17	290	0.13	1.7	1.8
25	35	1.4	0.34	540	0.26	1.6	1.9
25	40	1.6	0.21	760	0.21	1.5	1.4
25	45	2.5	0.40	960	0.22	1.4	1.3
25	50	3.2	0.60	1300	0.10	1.4	1.4
25	55	2.9	0.29	1700	0.27	1.3	1.3

Experimental Results

polygon-oriented			
$\|\|\mathbf{P}\|\|$	$\|\mathbf{P}\|$	μ	$\frac{\mu_n}{\mu_{n-1}}$
2000	5	0.021	-
2000	10	0.064	3.0
2000	15	0.2	3.1
2000	20	1.0	5.0
2000	25	5.2	5.2
2000	30	15	2.9
2000	35	49	3.3
2000	40	400	8.2

vertex-oriented			
$\|\|\mathbf{P}\|\|$	$\|\mathbf{P}\|$	μ	$\frac{\mu_n}{\mu_{n-1}}$
500	5	61	-
500	10	52	0.9 (!)
500	15	48	0.9 (!)
500	20	70	1.5
500	25	75	1.1
500	30	84	1.1
500	35	110	1.3
500	40	120	1.1

Future Work

- parallelizing the search
- approximation for larger instances
- consider other gripper shapes
- workspace limitations of the gripper

Acknowledgments

Thanks to

- Torben Hagerup
- Christian Rähtz
- Lev Sorokin

Covering Questions

Spørgsmål?

ερωτήσεις?

שאלות?

Questions?

Fragen?

Вопросов?

Pitanja?

Kérdések?

Perguntas?

Domande?

¿Preguntas?

Pytania?

Vraagen?

Sorulur?

Întrebari?