



**OPPA European Social Fund
Prague & EU: We invest in your future.**



DCGI

DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

ARRANGEMENTS (uspořádání)

PETR FELKEL

FEL CTU PRAGUE

felkel@fel.cvut.cz

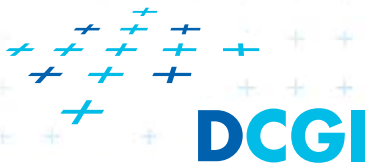
<http://service.felk.cvut.cz/courses/X36VGE>

Based on [Berg], [Mount]

Version from 16.12.2011

Talk overview

- Arrangements of lines
 - Incremental construction
 - Topological plane sweep



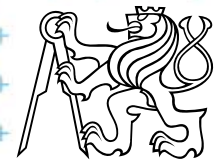
Line arrangement

- The next most important structure in CG after CH, VD, and DT
- Possible in any dimension
arrangement of $(d-1)$ -dimensional hyperplanes
- We concentrate on lines in the plane
- Defined on terms of set of lines
(set of points up to now) but
- Typical application is solving problems of point sets in dual plane

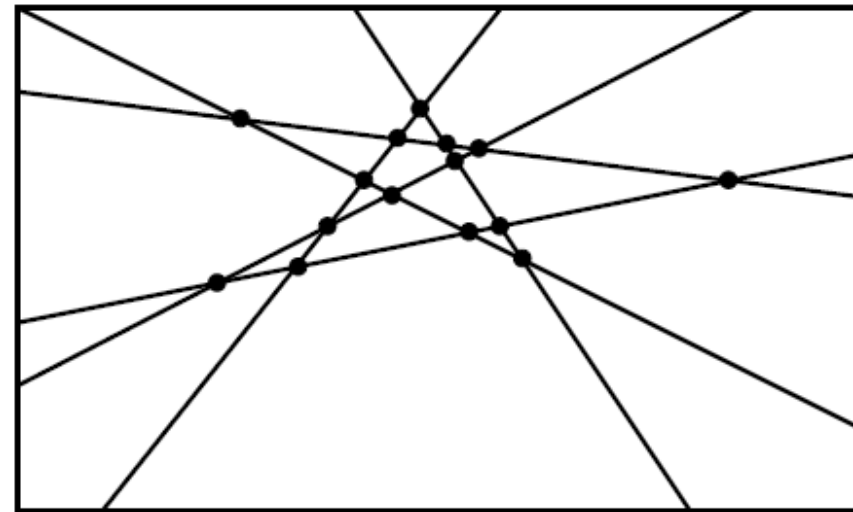
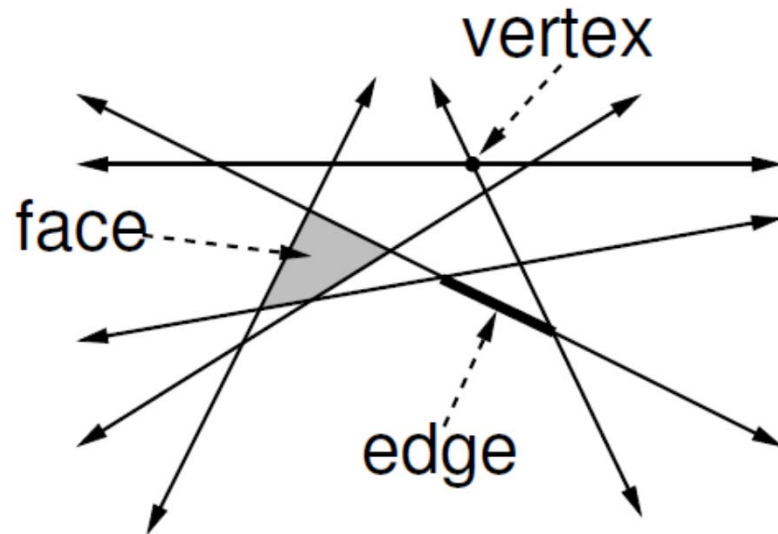


Line arrangement

- A finite **set L of lines** subdivides the plane into a cell complex, called **arrangement $A(L)$**
- Can be defined also for curves & surfaces...
- In plane, arrangement defines a **planar graph**
 - Vertices – intersections of lines (2 or more)
 - Edges – intersection free segments (or rays or lines)
 - Faces – convex regions containing no line (possibly unbounded)
- **Formal problem: graph must have bounded edges**
 - Topological fix: vertex in infinity
 - Geometrical fix: BBOX, often enough as abstract with corners $\{-\infty, -\infty\}, \{\infty, \infty\}$

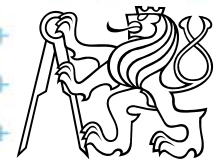


Line arrangement



bounding box [Mount]

- Simple arrangement assumption
 - = no three lines intersect in a single point
 - Careful implementation or symbolic perturbation



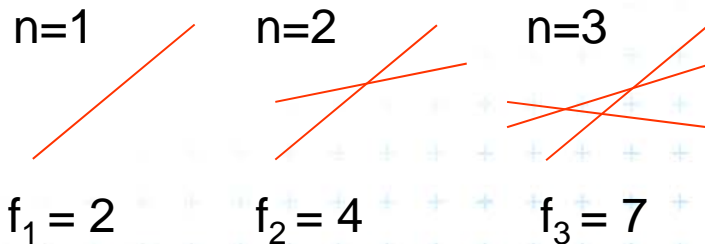
Combinatorial complexity of line arrangement

- $O(n^2)$
- Given n lines in general position, max numbers are
 - Vertices $\binom{n}{2} = \frac{n(n-1)}{2}$ - each line intersect $n - 1$ others
 - Edges n^2 - $n-1$ intersections create n edges on each of n lines

– Faces $\frac{n(n+1)}{2} + 1 = \binom{n}{2} + n + 1$

$$f_0 = 1$$

$$f_n = f_{n-1} + n$$



$$f_n = f_0 + \sum_{i=1}^n i = \frac{n(n+1)}{2} + 1$$



Construction of line arrangement

(0. Plane sweep method)

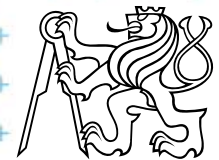
- $O(n^2 \log n)$ time and $O(n)$ storage plus $O(n^2)$ storage for the arrangement ($\log n$ - heap access, n^2 vertices, edges, faces)

1. Incremental method

- $O(n^2)$ time and $O(n^2)$ storage
- Optimal method

2. Topological plane sweep

- $O(n^2)$ time and $O(n)$ storage only
- Does not store the result arrangement
- Useful for applications that may throw the arrangement away after processing



1. Incremental construction of arrangement

- $O(n^2)$ time, $O(n^2)$ space
~size of arrangement \Rightarrow it is an optimal algorithm
- Not randomized – depends on n only, not on order
- Add line l_i one by one ($i = 1 .. n$)
 - Find the leftmost intersection with BBOX
among $2(i-1)+4$ edges on the BBOX ... $O(i)$
 - Trace the line through the arrangement $A(L_{i-1})$ and split
the intersected faces ... $O(i)$ – why? See later
 - Update the subdivision (cell split) ... $O(1)$
- Altogether $O(n^2)$



1. Incremental construction of arrangement

Arrangement(L)

Input: Set of lines L in general position (no 3 intersect in 1 common point)

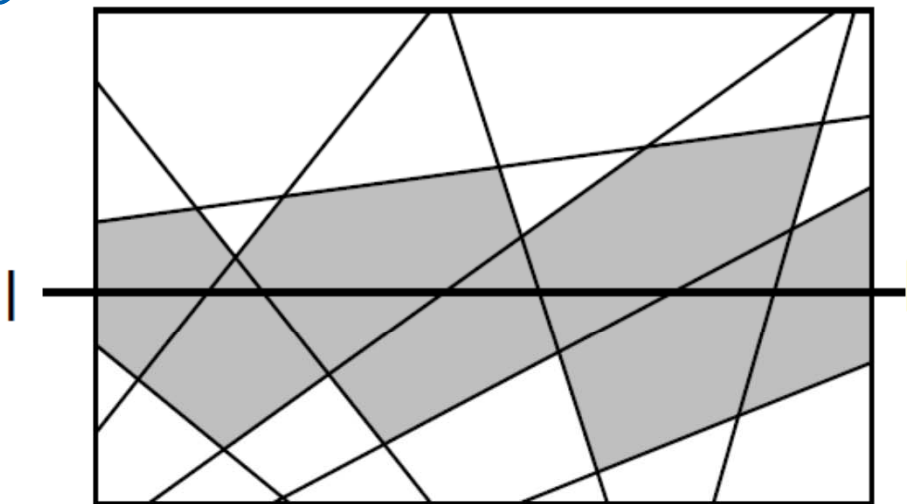
Output: Line arrangement $A(L)$ (resp. part of the arrangement stored in BBOX $B(L)$ containing all the vertices of $A(L)$)

1. Compute the BBOX $B(L)$ containing all the vertices of $A(L)$... $O(n^2)$
2. Construct DCEL for the subdivision induced by $B(L)$... $O(1)$
3. **for** $i = 1$ **to** n **do** // *insert line* l_i
4. find edge e , where line l_i intersects the BBOX of $2(i-1)+4$ edges ... $O(i)$
5. $f =$ bounded face incident to e
6. **while** f is in $B(L)$ ($f =$ bounded face – in the BBOX) ... $O(???)$
7. split f and set f to be the next intersected face
8. update the DCEL (split the cell) ... $O(1)$



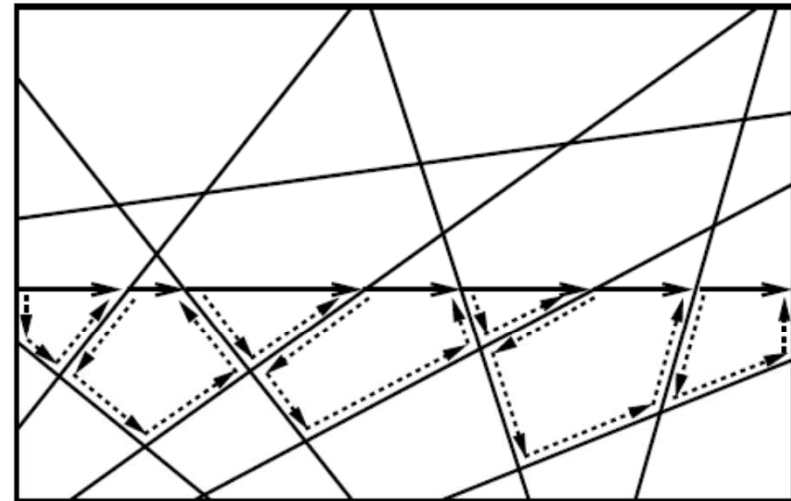
Tracing the line through the arrangement

- Walk around edges of current face (face walking)
- Determine if the line l_i intersects this edge
- When intersection found, jump to the face on the other side of this edge



The zone of l_i

$n=8$ lines, 7 faces in the zone, 22 edges tested of max 48



Walking the lower part
of the zone



Tracing the line through the arrangement

- Number of traversed edges determines the insertion complexity
- Naïve estimation would be $O(i^2)$ traversed edges (i faces, i lines per face, i^2 edges)
- According to the Zone theorem, it is $O(i)$ edges only!

Zone theorem

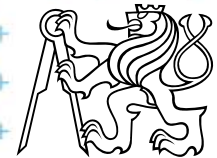
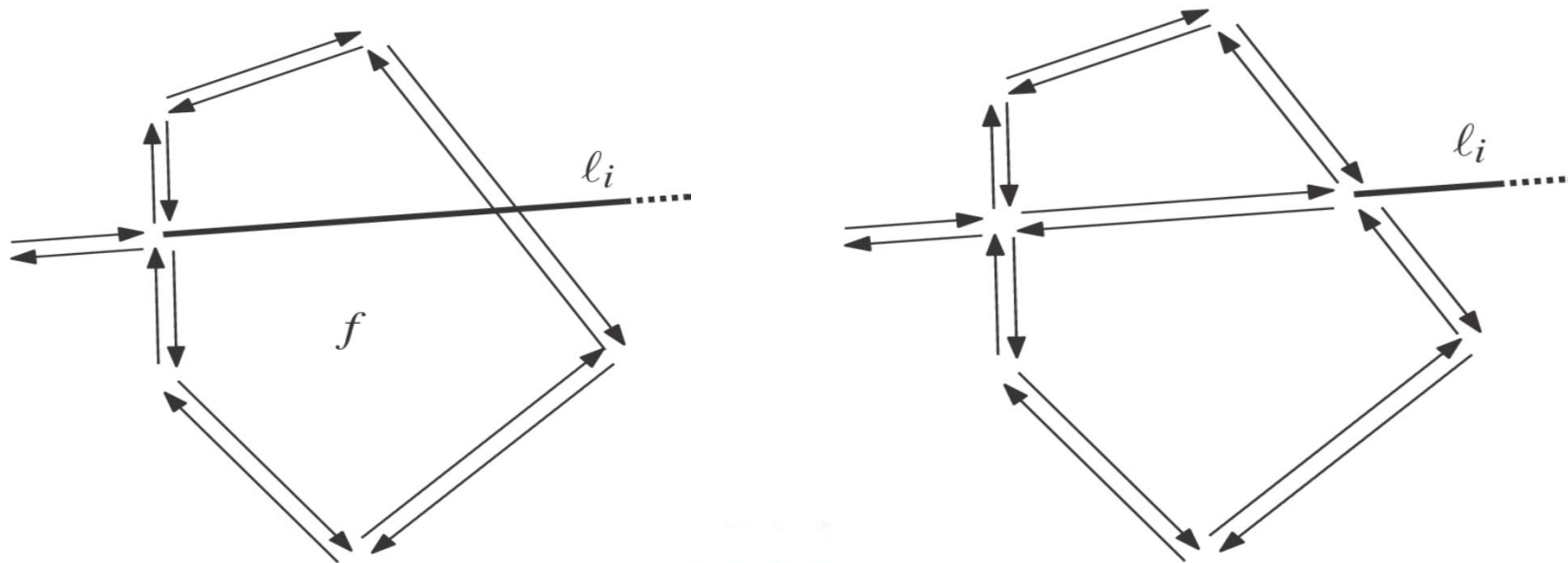
= given an arrangement $A(L)$ of n lines in the plane and given any line l in the plane, the total number of edges in all the cells of the zone $Z_A(l)$ is at most $6n$.

For proof see [Mount, page 69]



Cell split

- 2 new face records, 1 new vertex, 2+2 new half-edges + update pointers ... $O(1)$



Complexity of incremental algorithm

- n insertions
- $O(i) = O(n)$ time for one line insertion
(Zone theorem)

=> Complexity: $O(n^2) + n \cdot O(i) = O(n^2)$

bbox edges walked



2. Topological plane sweep algorithm

- Complete arrangement needs $O(n^2)$ storage
- Often we need just to **process each arrangement element just once** – and we can throw it then
- Classical **Sweep line** algorithm
 - needs $O(n)$ storage
 - needs $\log n$ for **heap** manipulation in $O(n^2)$ event points
 $\Rightarrow O(n^2 \log n)$ algorithm
- **Topological sweep line - TSL**
 - disperses $O(\log n)$ factor in time
 - **array** of neighbors and a **stack** of ready vertices
 $\Rightarrow O(n^2)$ algorithm

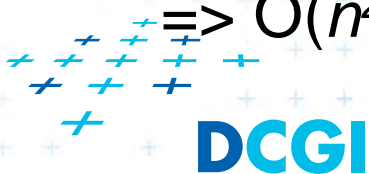
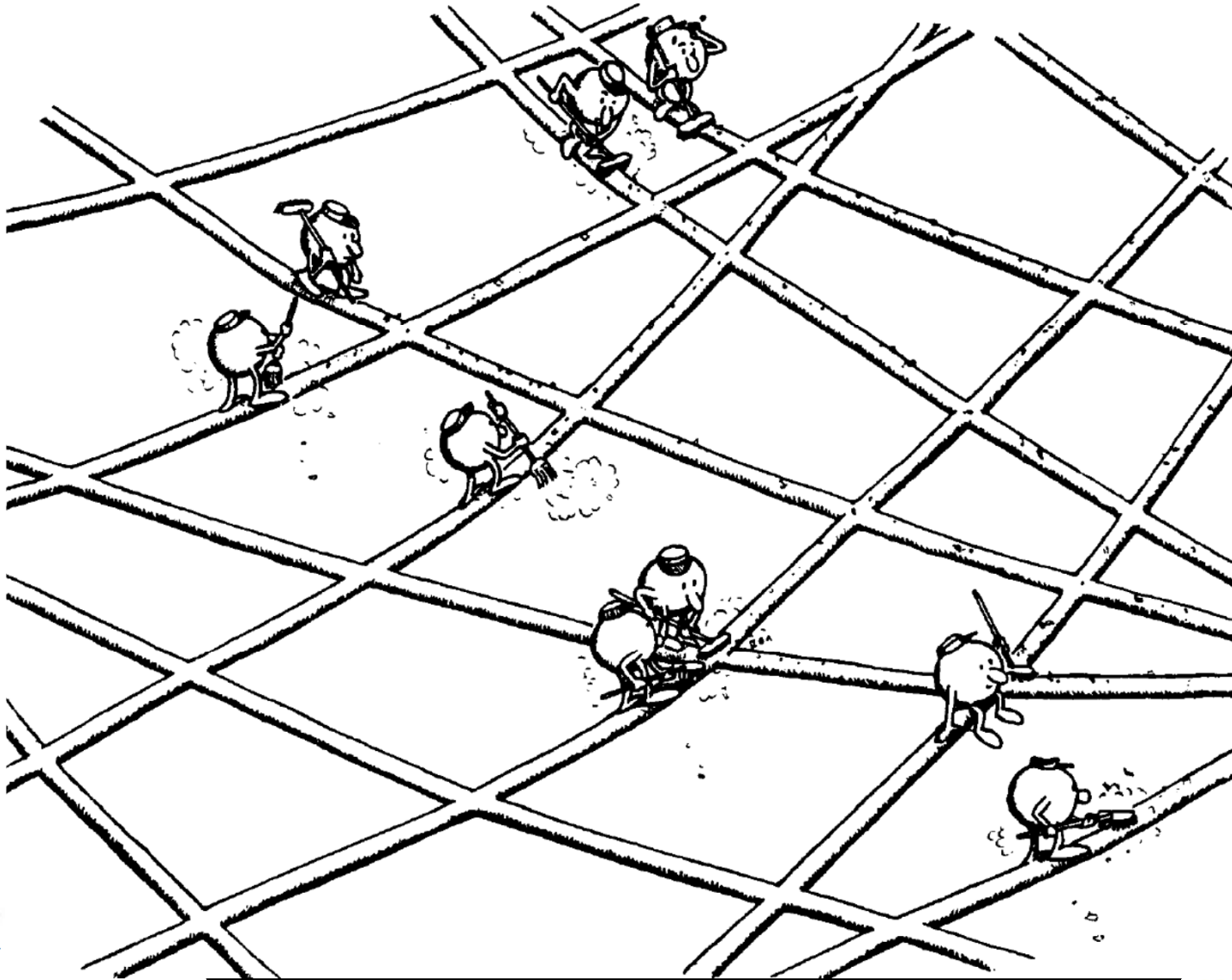


Illustration from Edelsbrunner & Guibas

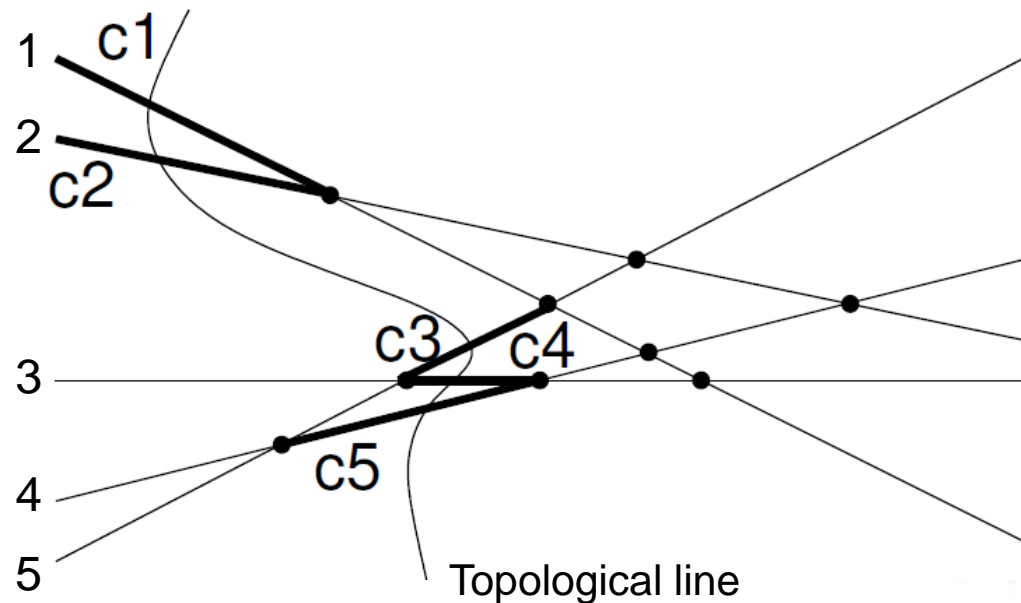


Topological line and cut

Topological line (curve)

(an intuitive notion)

- Monotonic line in y-dir
- intersects each line exactly once (as a sweep line)



Cut in an arrangement A

- is a sequence of edges c_1, c_2, \dots, c_n in A (one taken from each line), such that for $1 \leq i \leq n-1$, c_i and c_{i+1} are **incident to the same face** of A and c_i is **above** and c_{i+1} **below** the face
- Edges not necessarily connected



Topological plane sweep algorithm

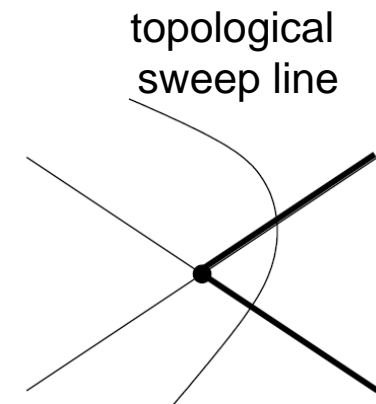
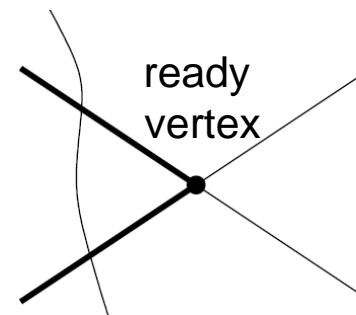
- Starts at the **leftmost cut**
 - Consist of left-unbounded edges of A (ending at $-\infty$)
 - Computed in $O(n \log n)$ time – inverse order of slopes
- The sweep line is
 - pushed from the leftmost cut to the rightmost cut
 - Advances in elementary steps

- **Elementary step**

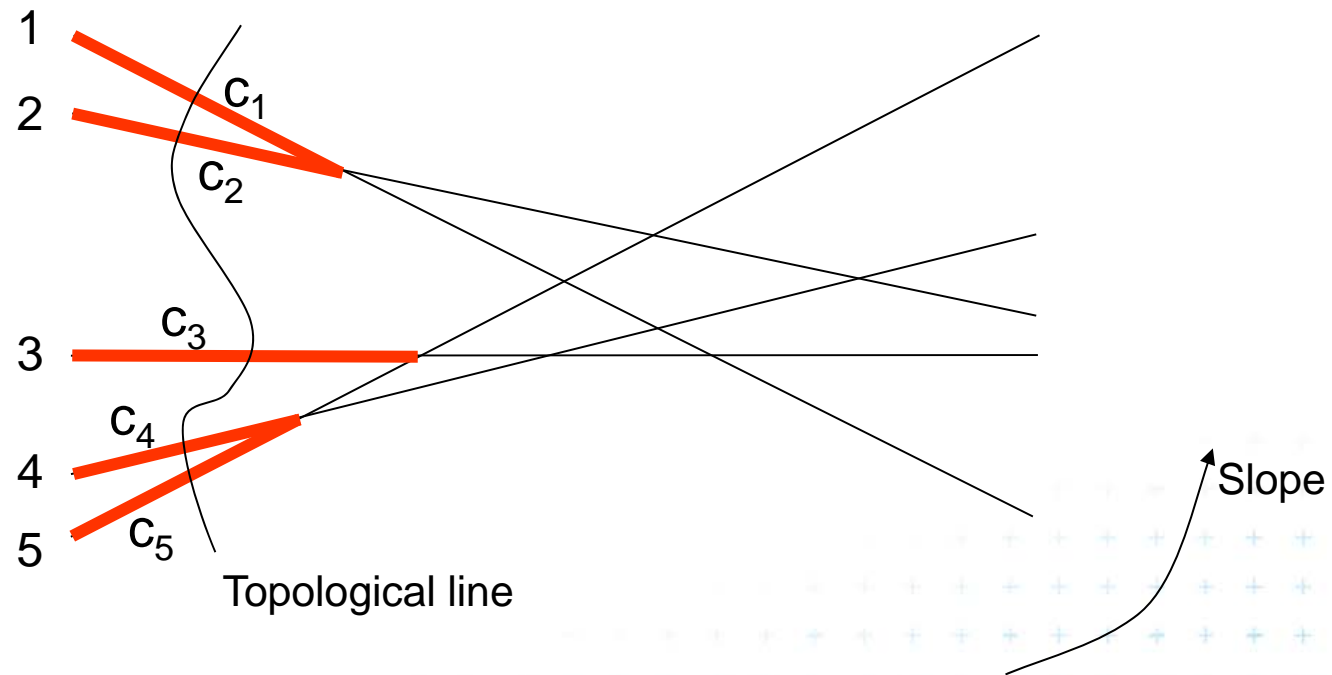
= Processing of a *ready vertex*

(intersection of consecutive edges at their right-point)

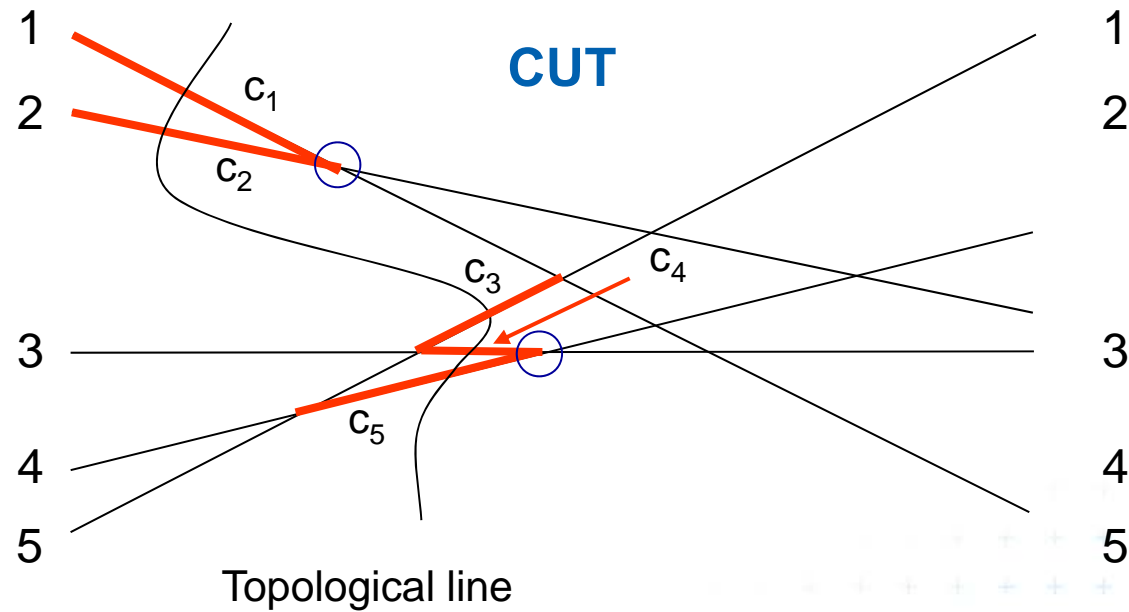
- Swaps the order of lines along the sweep line
- Is always possible (e.g., the point with smallest x)
- Searching of smallest x would need $O(\log n)$ time



The leftmost cut

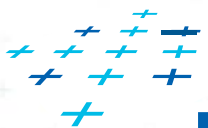
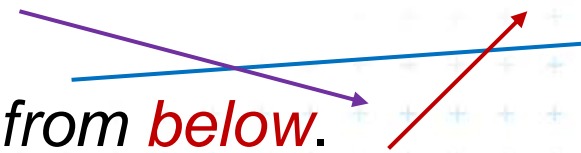


The cut during the topological plane sweep



How to determine the next right point?

- **Elementary step** (intersection at edges right-point)
 - Is always possible (e.g., the point with smallest x)
 - But searching the smallest x would need $O(\log n)$ time
 - We need $O(1)$ time
- **Right endpoint** of the edge in the cut results from
 - a line of *smaller slope* intersecting it *from above* (traced from L to R) or
 - line of *larger slope* intersecting it *from below*.
- **Use Upper and Lower Horizon Trees (UHT, LHT)**
 - Common segments of UHT and LHT belong to the cut
 - Intersect the trees, find pairs of consecutive edges
 - use the right points as legal steps (push to stack)



Upper and lower horizon tree

- Upper horizon tree (UHT)

- Insert lines in order of **decreasing** slope
- When two edges meet, **keep the edge with higher slope and trim the edge with lower slope**
- To get one tree and not the forest of trees (if not connected) add vertical line in $+\infty$
- **Left endpoints** of the edges in the cut do not belong to the tree



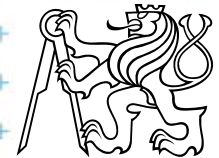
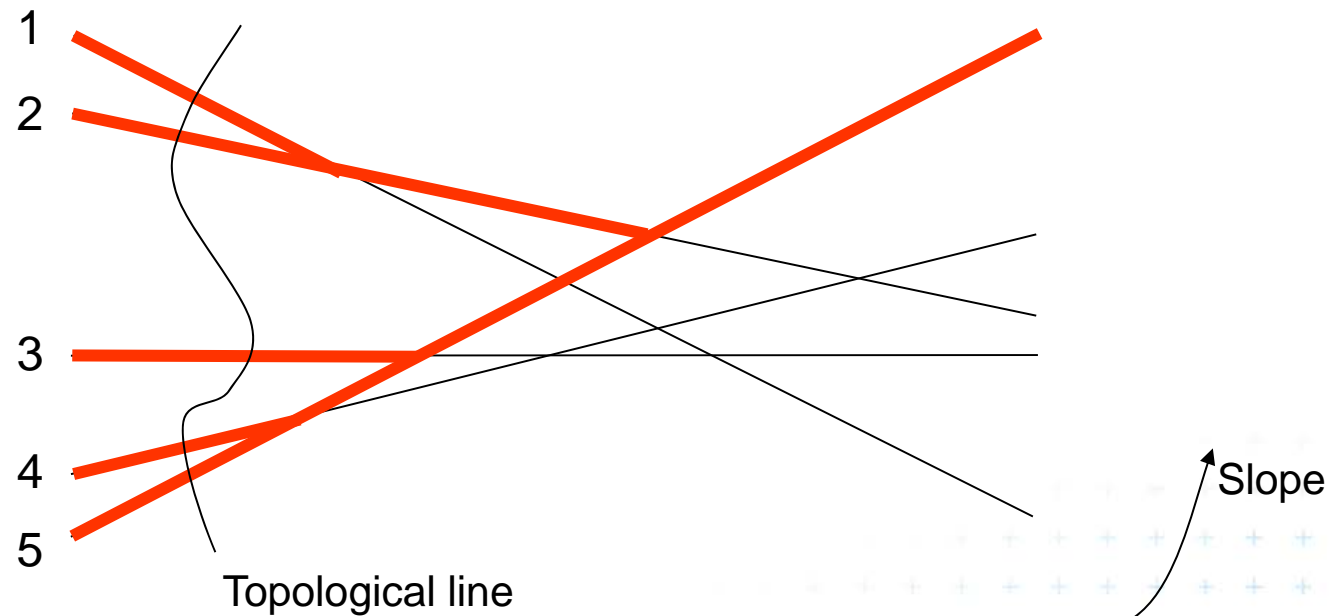
- Lower horizon tree (LHT) is symmetrical

- UHT and LHT **serve for right endpoints determination**



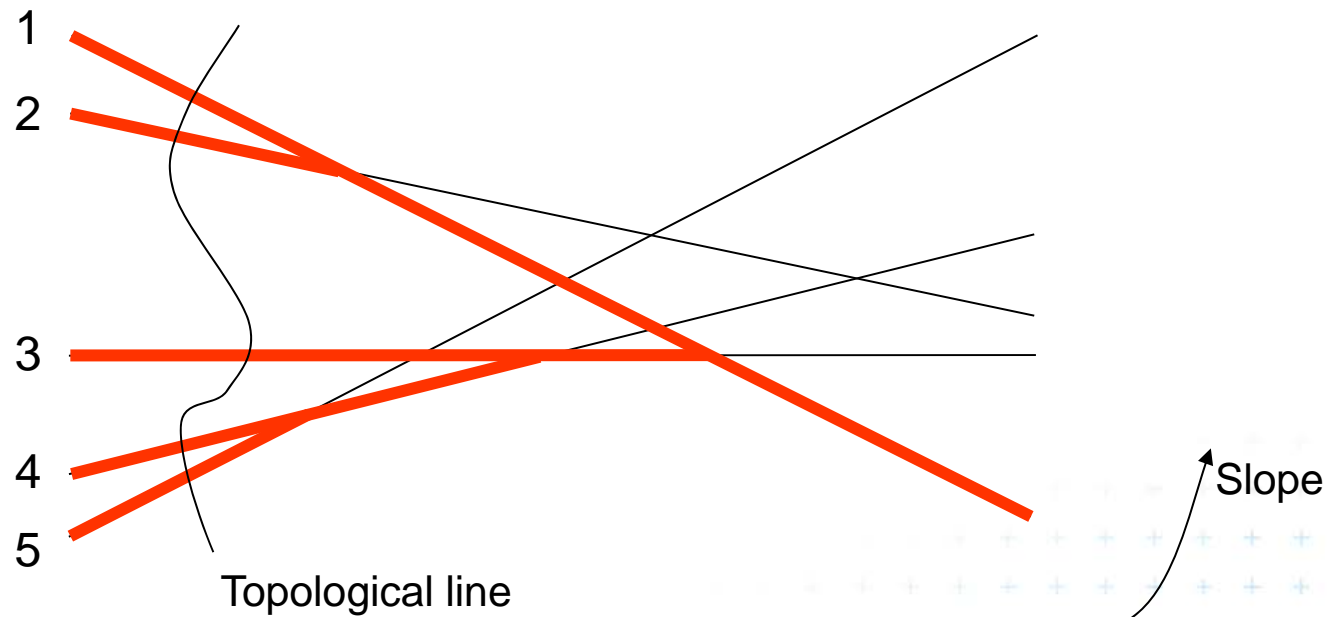
Upper horizon tree (UHT) – initial tree

- Insert lines in order of **decreasing slope**



Lower horizon tree (LHT) – initial tree

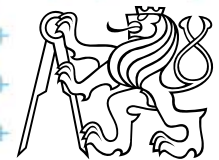
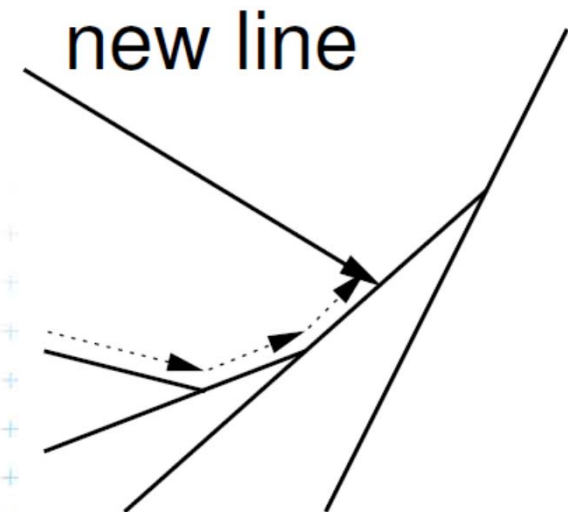
- Insert lines in order of **increasing slope**



Upper horizon tree (UHT) – init. construction

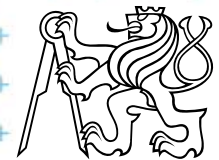
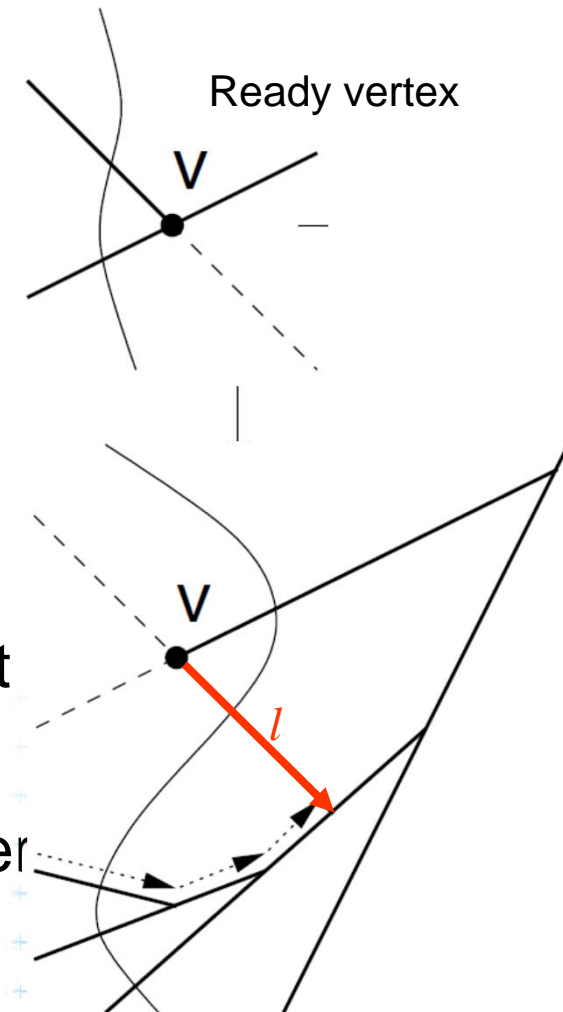
- Insert lines in order of **decreasing slope**
- Each new line starts above all the current lines
- The uppermost face = convex polygonal chain
- Walk left to right along the chain to determine the intersection
- Never walk twice over segment
 - Such segment is no longer part of the upper chain
 - $O(n)$ segments in UHT

$\Rightarrow O(n)$ initial construction
(after $n \log n$ sorting of the lines)



Upper horizon tree (UHT) – update

- After the elementary step
- Two edges swap position along the sweep line
- Lower edge l
 - Reenter to UHT
 - Terminate at nearest edge of UHT
 - Start in edge below in the current cut
 - Traverse the face in CCW order
 - Intersection must exist, as l has lower slope than the other edge from v and both belong to the same face



Data structures for topological sweep alg.

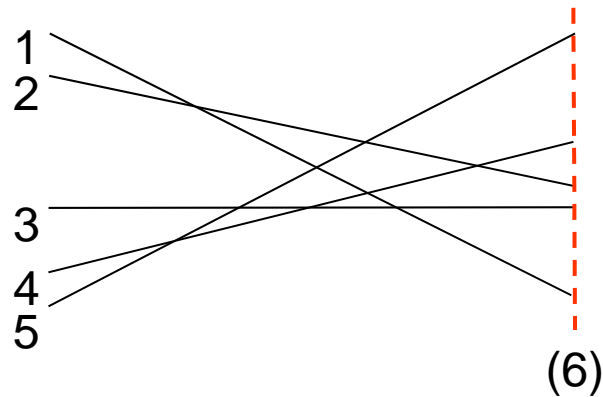
Topological sweep line algorithm uses 5 arrays:

1. Line equation coefficients – $E [1:n]$
2. Upper horizon tree – UHT $[1:n]$
3. Lower horizon tree – LHT $[1:n]$
4. Order of lines cut by the sweep line – $C [1:n]$
5. Edges along the sweep line – $N [1:n]$
6. Stack for ready vertices (events) – S

(n number of lines)



1) Line equation coefficients $E [1:n]$



- Array of line equation coefs. E
 - Contains coefficients a_i and b_i of line equations $y = a_i x + b_i$
 - E is indexed by the **line index**
 - **Lines are ordered** according to their slope (angle from -90° to 90°)

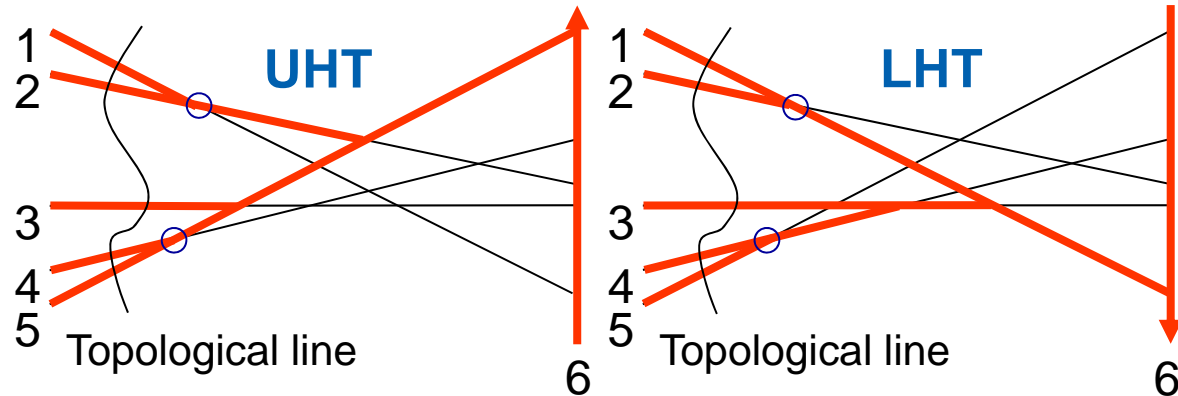
Array of line equations E
 $y = a_i x + b_i$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5



2) and 3) – Horizon trees UHT and LHT

Their intersection is used for searching of legal steps (right points)
 - the shorter edge wins



UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

Store pairs of line indices in E that delimit segment l_i to the left and to the right

Unlimited line has “indices” $[-\infty, +\infty]$

One additional vertical line

- prevents the tree from splitting into forest of trees
- is inserted first and never trimmed
- is $[-\infty, +\infty]$ for UHT
- is $[\infty, -\infty]$ for LHT

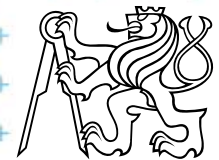


4) Order of lines cut by sweep line – $C [1:n]$

- The topological sweep line cuts each line once
- Order of these cuts (along the topological sweep line) is stored in array C as a sequence of line indices
- For the initial leftmost cut, the order is the same as in E
- Index c_i addresses i -th line from top along the sweep line

CUT Lines C
Indexes of supporting lines

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5



5) Edges along the sweep line – $N [1:n]$

- Edges intersected by the topological sweep line are stored here (edges along the sweep line)
- Instead of endpoints themselves, we store the **indices of lines whose intersections delimit the edge**
- Order of these edges is the same as in C (this is the way used in the original paper)
- Index c_i addresses i -th edge from top along the sweep line

CUT edges N
Pairs of line indices
delimiting the edge

c_1	$-\infty$	2
c_2	$-\infty$	1
c_3	$-\infty$	5
c_4	$-\infty$	5
c_5	$-\infty$	4



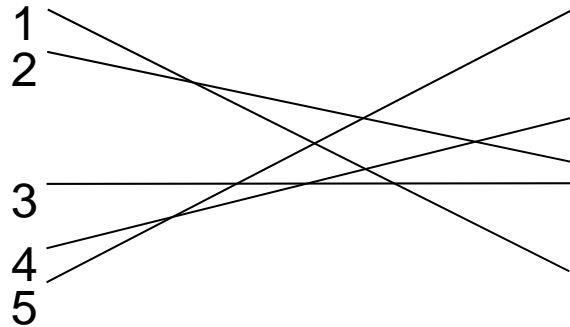
6) Stack S

- The exact order of events is not important
- Alg. can process any of the “ready vertex”
- **Event queue** is therefore **replaced by a stack** (faster – $O(1)$ instead of $O(\log n)$ of the queue)
- The stack stores just the **upper edge c_i**
- Intersection in the ready vertex is computed between stored c_i and c_{i+1}

Stack S
Ready vertex
first edge idx



Topological sweep line demo



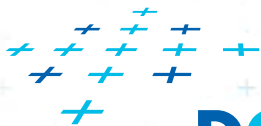
Array of line equations E

$$y = a_i x + b$$

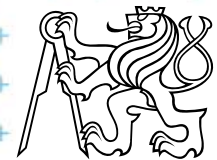
1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

Input

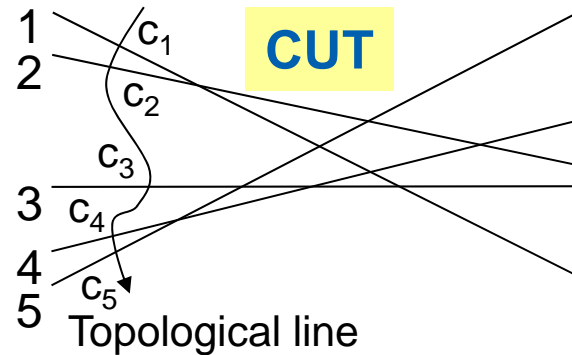
- set of lines L in the plane
- ordered in increasing slope (-90° to 90°), simple, not vertical
- line parameters in array E



DCGI



1. Initial leftmost cut - C



- Store the line indices into the Cut lines array C in increasing slope order

Array of line equations E
 $y = a_i x + b$

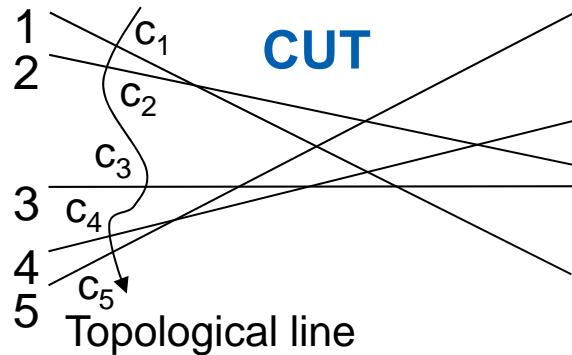
1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

CUT Lines C
Indexes of supporting lines

c1	1
c2	2
c3	3
c4	4
c5	5



1. Initial leftmost cut - N



- Prepare array N for endpoints of the cutted edges (resp. for line indices delimiting these edges)

Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

indices of lines



DCGI

CUT edges N
 Pairs of line indices delimiting the edge

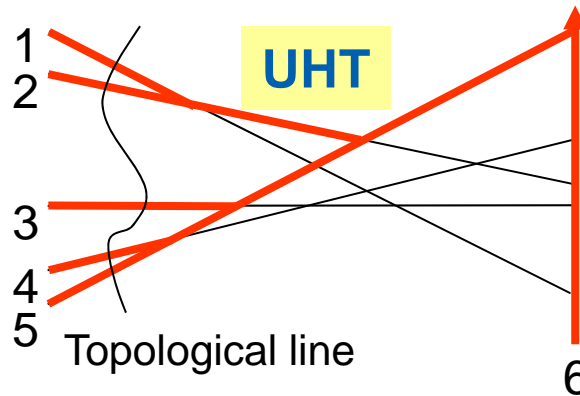
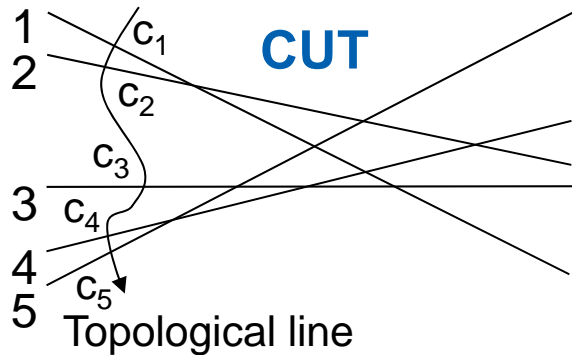
c1	$-\infty$	∞
c2	$-\infty$	∞
c3	$-\infty$	∞
c4	$-\infty$	∞
c5	$-\infty$	∞

CUT Lines C
 Indexes of supporting lines

c1	1
c2	2
c3	3
c4	4
c5	5



2a) Compute Upper Horizon Tree - UHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

CUT edges N
Pairs of line indices delimiting the edge

c_1	$-\infty$	∞
c_2	$-\infty$	∞
c_3	$-\infty$	∞
c_4	$-\infty$	∞
c_5	$-\infty$	∞

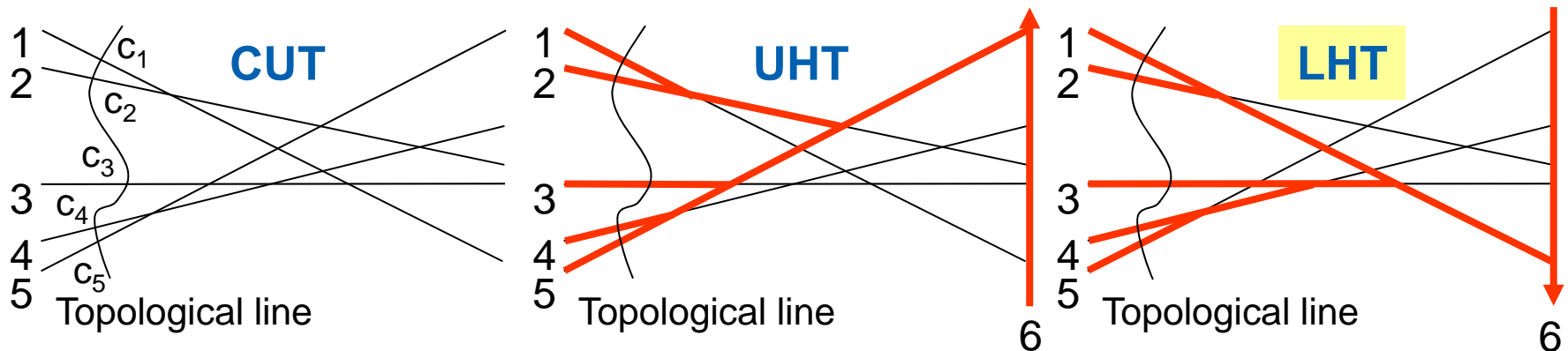
CUT Lines C
Indexes of supporting lines

c_1	1
c_2	2
c_3	3
c_4	4
c_5	5

Inserted first, never changed



2b) Compute Lower Horizon Tree - LHT



Array of line equations E
 $y = a_i x + b_i$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c1	$-\infty$	∞
c2	$-\infty$	∞
c3	$-\infty$	∞
c4	$-\infty$	∞
c5	$-\infty$	∞

CUT Lines C
 Indexes of supporting lines

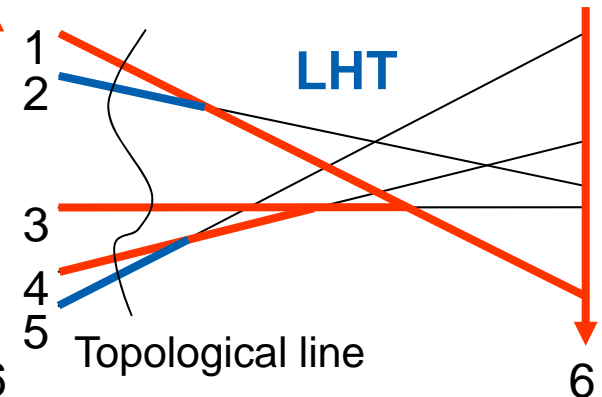
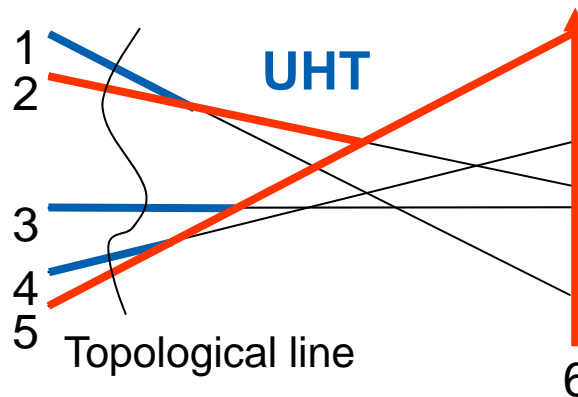
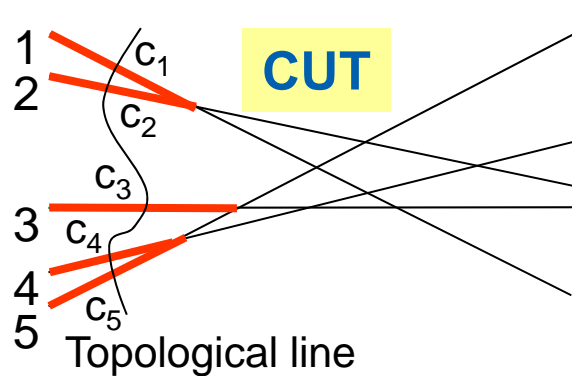
c1	1
c2	2
c3	3
c4	4
c5	5

Stack S
 Ready vertex first edge idx

Inserted first, never changed



3a) Determine right delimiters of edges - N



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

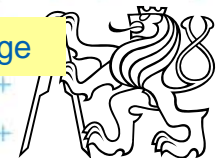
c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	$-\infty$	5
c5	$-\infty$	4

CUT Lines C
Indexes of supporting lines

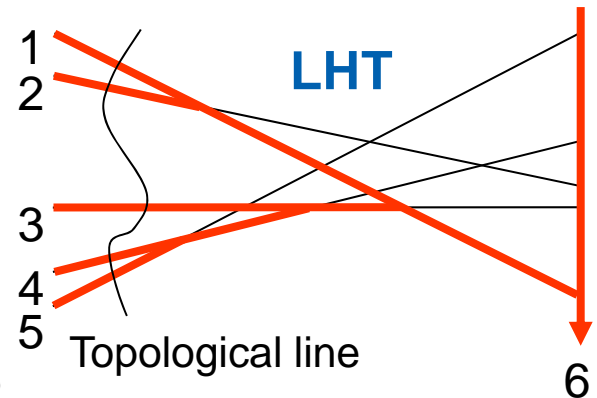
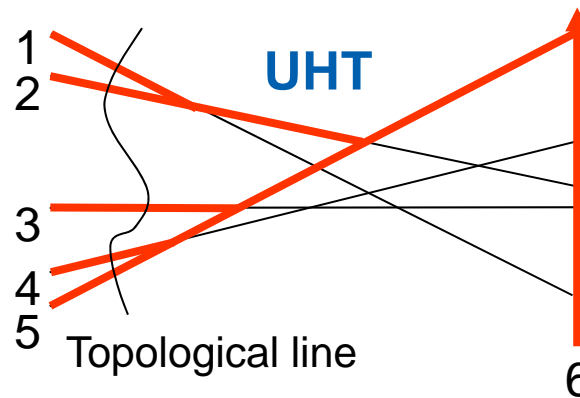
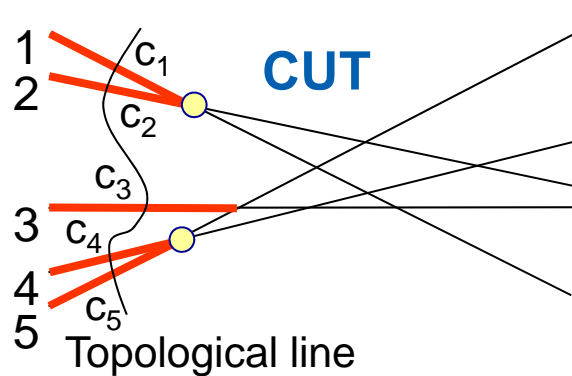
c1	1
c2	2
c3	3
c4	4
c5	5

Stack S
Ready vertex first edge idx

Intersect the trees – take the shorter edge



3b) Ready vertices = int. of neighbors – S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

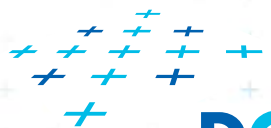
c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	$-\infty$	5
c5	$-\infty$	4

CUT Lines C
Indexes of supporting lines

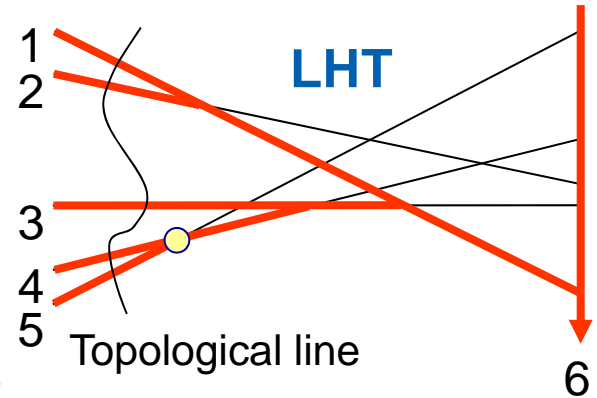
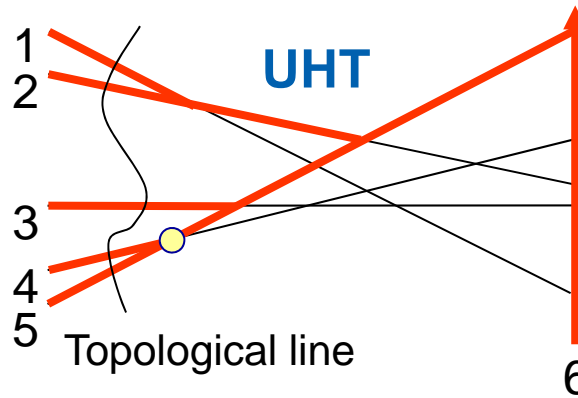
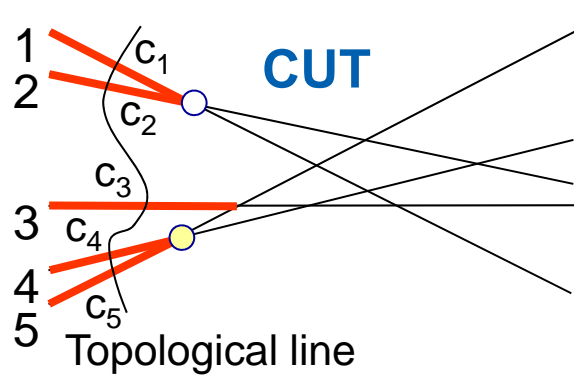
c1	1
c2	2
c3	3
c4	4
c5	5

Stack S
Ready vertex first edge idx

c4
c1



4a) Pop ready vertex from S – process c4



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

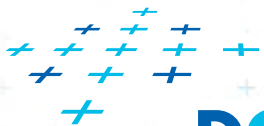
c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	$-\infty$	5
c5	$-\infty$	4

CUT Lines C
Indexes of supporting lines

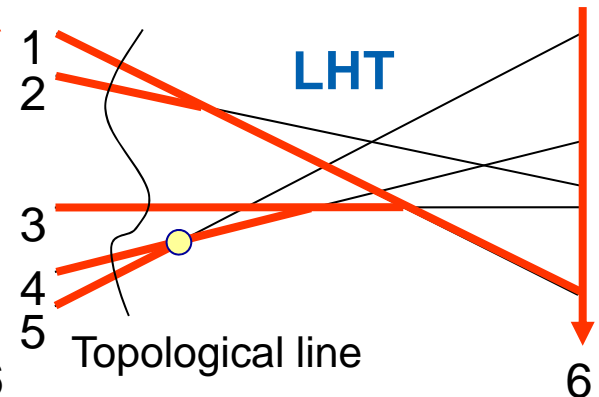
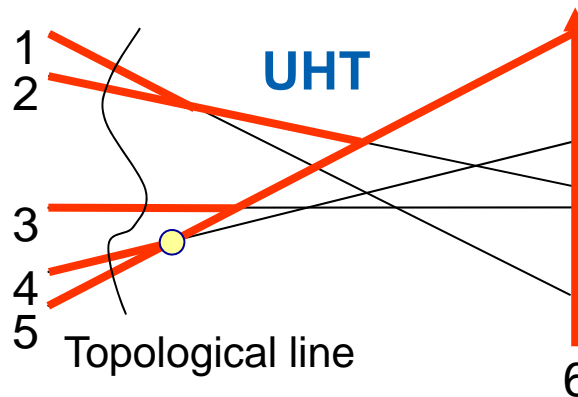
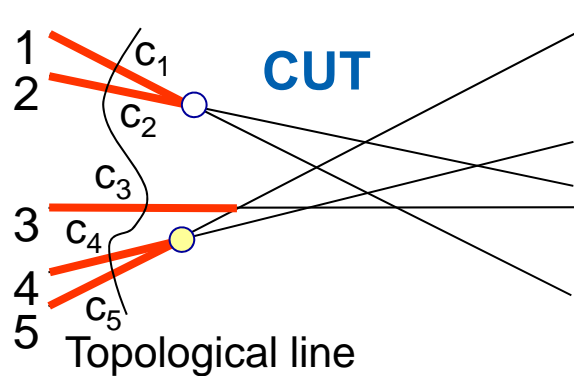
c1	1
c2	2
c3	3
c4	4
c5	5

Stack S
Ready vertex first edge idx

c4
c1



4b) Swap lines c4 and c5 – swap 4 and 5



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	$-\infty$	5
5	$-\infty$	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	$-\infty$	3
5	$-\infty$	4
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	$-\infty$	4
c5	$-\infty$	5

CUT Lines C
Indexes of supporting lines

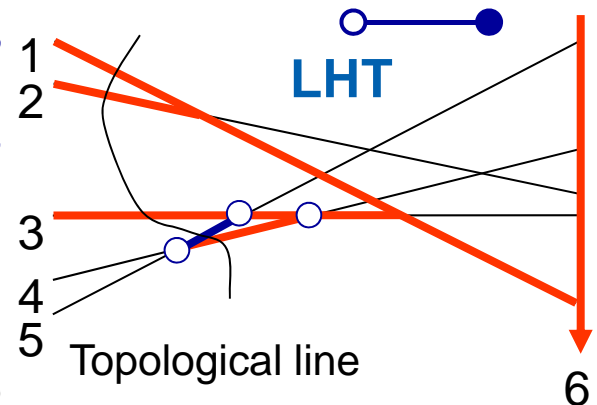
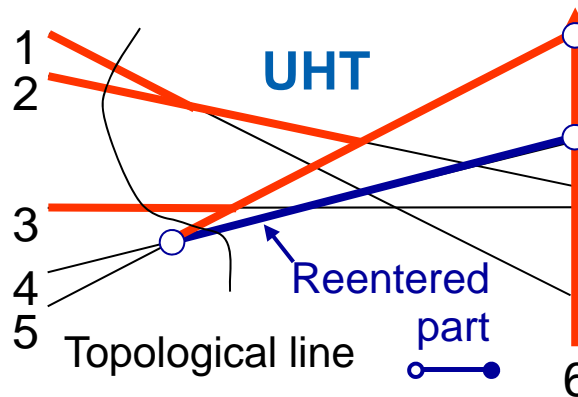
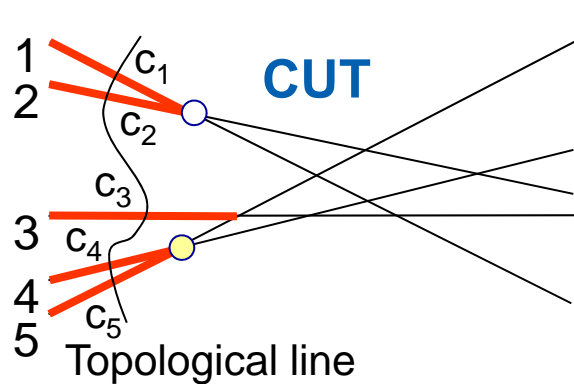
c1	1
c2	2
c3	3
c4	5
c5	4

Stack S
Ready vertex first edge idx

c1



4c) Update the horizon trees – UHT and LHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

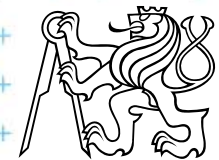
c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	$-\infty$	4
c5	$-\infty$	5

CUT Lines C
 Indexes of supporting lines

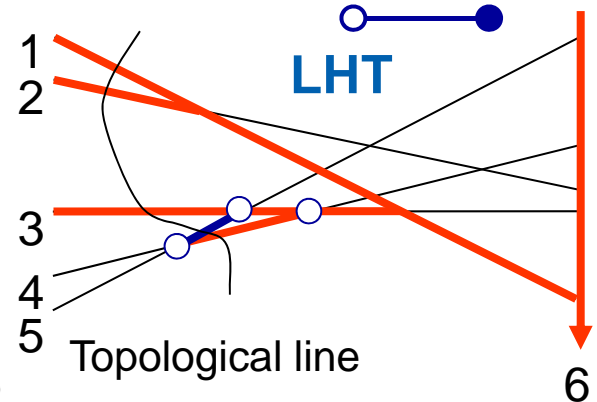
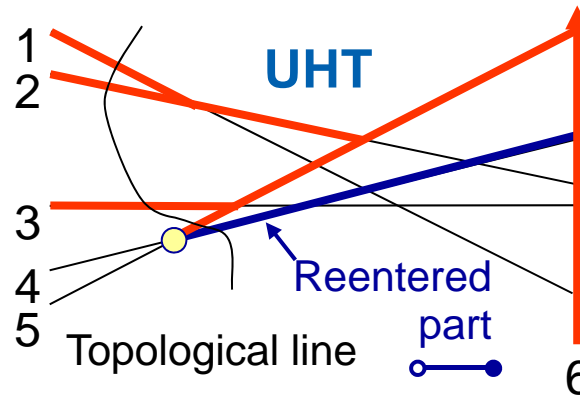
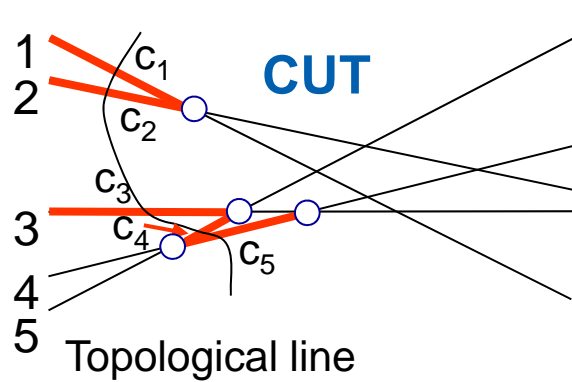
c1	1
c2	2
c3	3
c4	5
c5	4

Stack S
 Ready vertex upper edge id

c1



4d) Determine new cut edges endpoints – N



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	4	3
c5	5	3

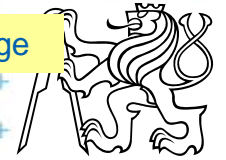
CUT Lines C
 Indexes of supporting lines

c1	1
c2	2
c3	3
c4	5
c5	4

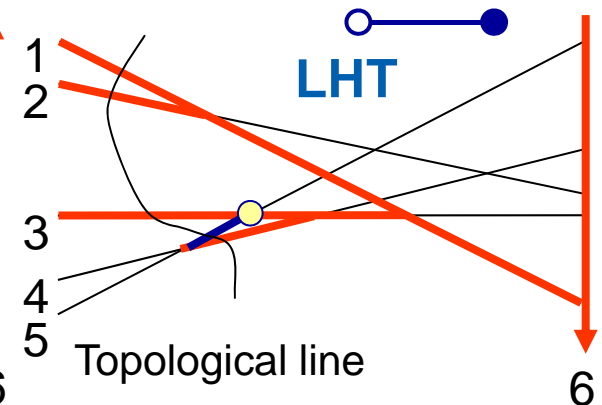
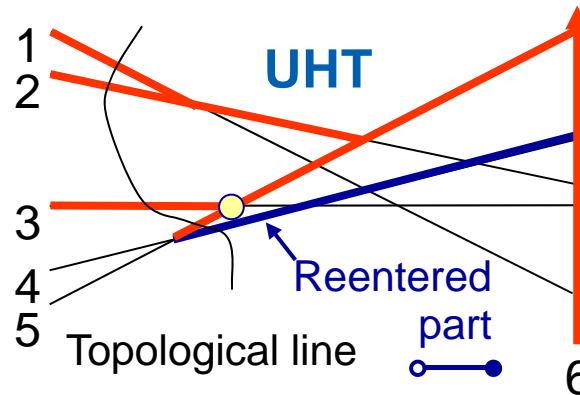
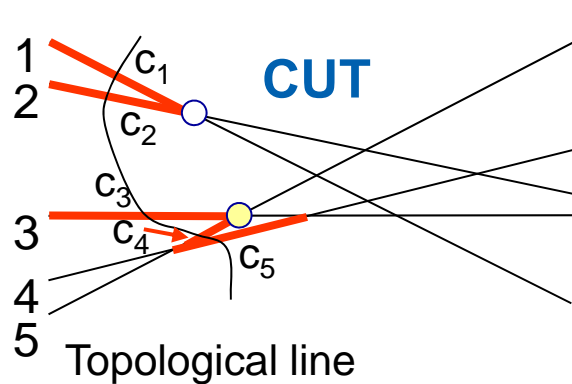
Stack S
 Ready vertex upper edge id

c1

Intersect the trees – take the shorter edge



4e) Intersect with neighbors – push into S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	4	3
c5	5	3

CUT Lines C
Indexes of supporting lines

c1	1
c2	2
c3	3
c4	5
c5	4

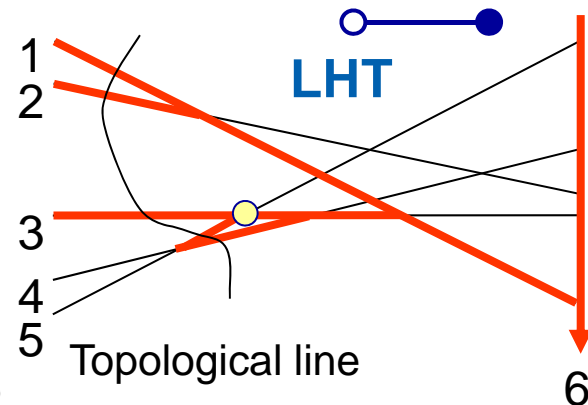
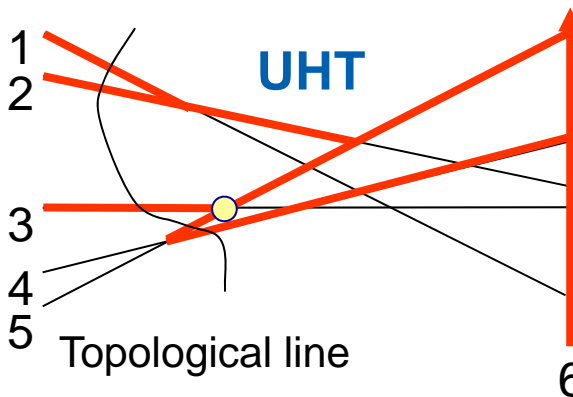
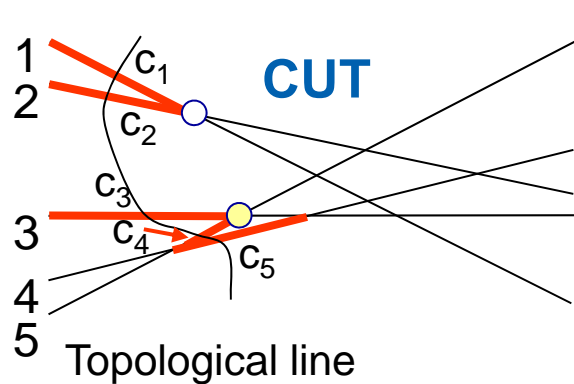
Stack S
Ready vertex upper edge id

c3
c1

Intersections of neighbors - into stack



4a) Pop ready vertex from S – process c3



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	$-\infty$	5
c4	4	3
c5	5	3

CUT Lines C
Indexes of supporting lines

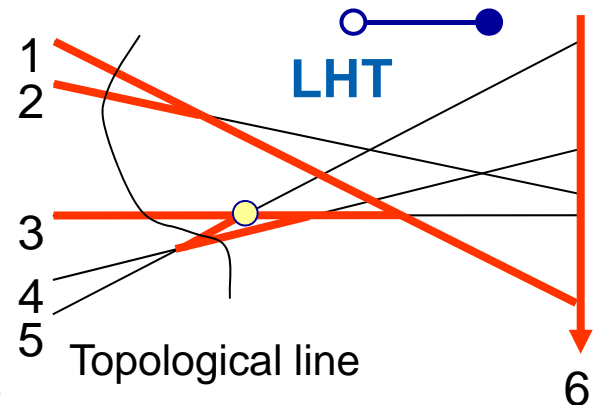
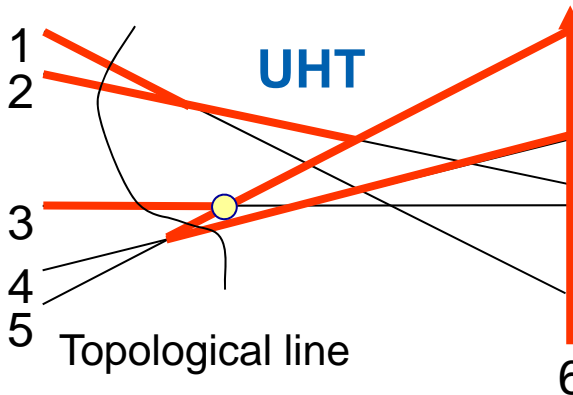
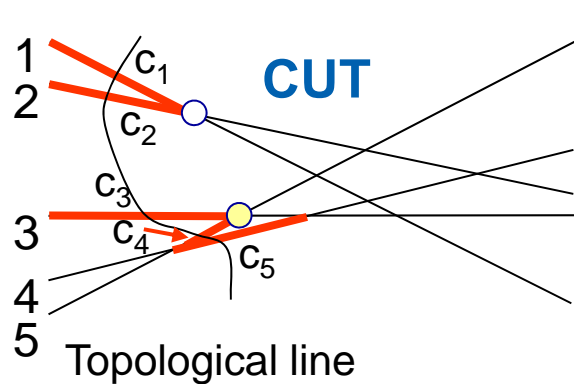
c1	1
c2	2
c3	3
c4	5
c5	4

Stack S
Ready vertex first edge idx

c3
c1



4b) Swap lines c4 and c5 – swap 4 and 5



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	$-\infty$	5
4	5	6
5	4	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	$-\infty$	1
4	5	3
5	4	3
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	4	3
c4	$-\infty$	5
c5	5	3

CUT Lines C
Indexes of supporting lines

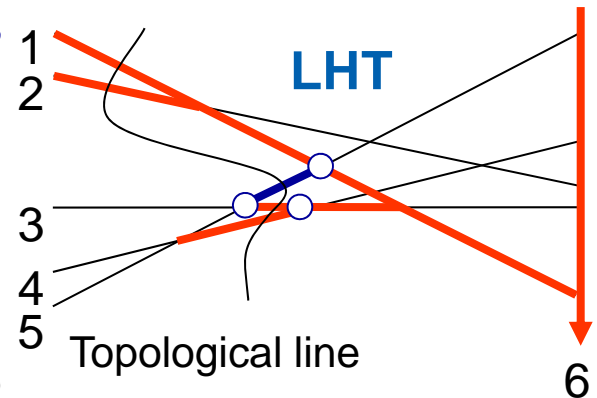
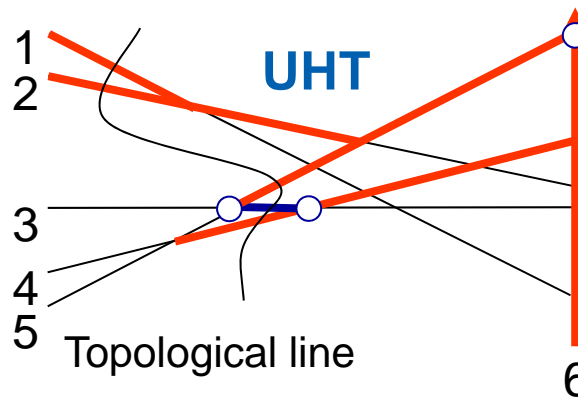
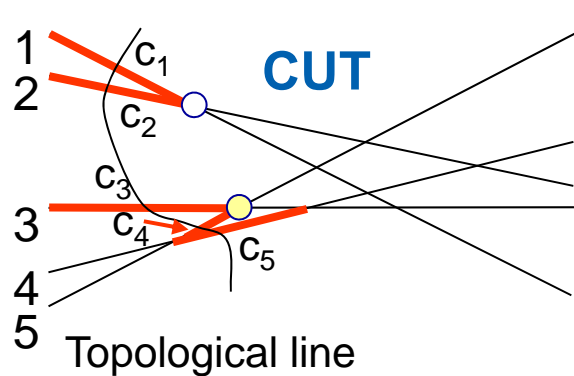
c1	1
c2	2
c3	5
c4	3
c5	4

Stack S
Ready vertex first edge idx

c3
c1



4c) Update the horizon trees – UHT and LHT



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
 Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
 Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
 Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	4	3
c4	$-\infty$	5
c5	5	3

CUT Lines C
 Indexes of supporting lines

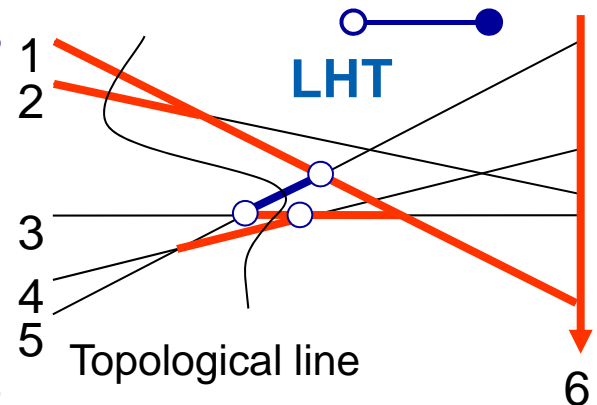
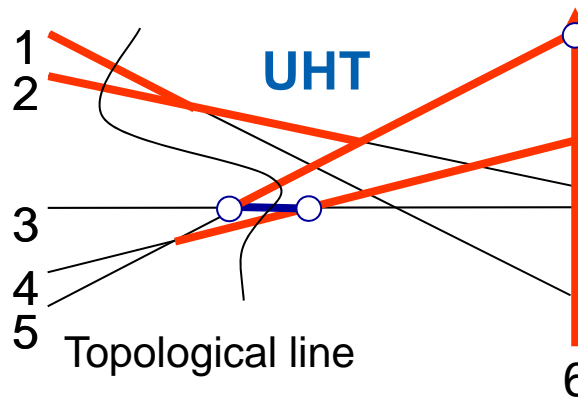
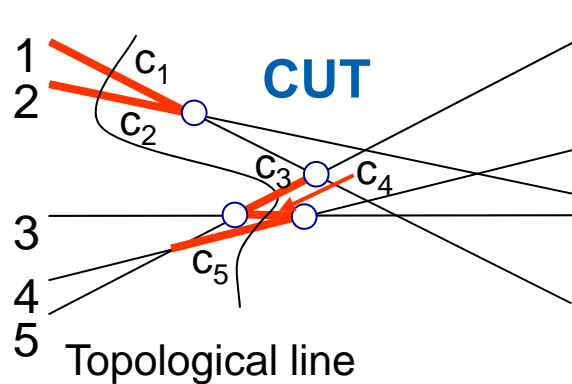
c1	1
c2	2
c3	5
c4	3
c5	4

Stack S
 Ready vertex first edge idx

c3
c1



4d) Determine new cut edges endpoints



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

c1	$-\infty$	2
c2	$-\infty$	1
c3	3	1
c4	5	4
c5	5	3

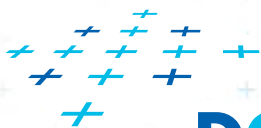
CUT Lines C
Indexes of supporting lines

c1	1
c2	2
c3	5
c4	3
c5	4

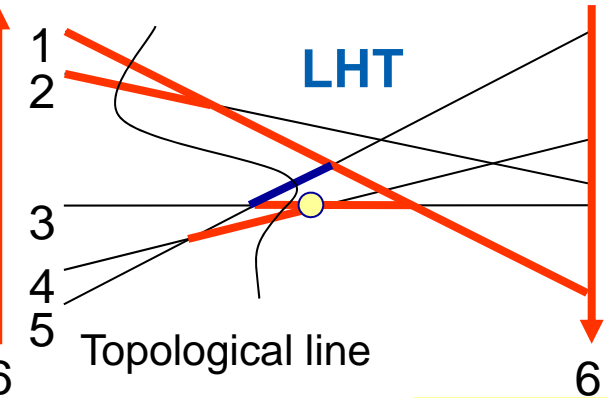
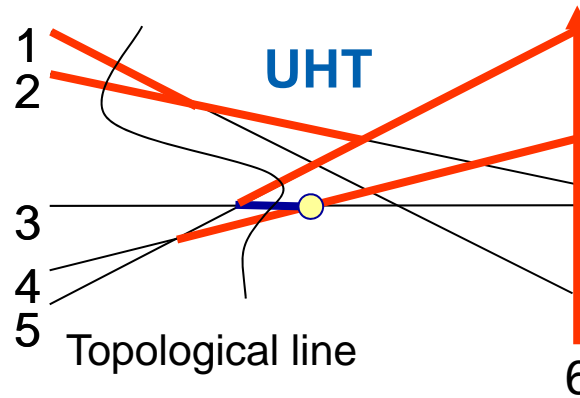
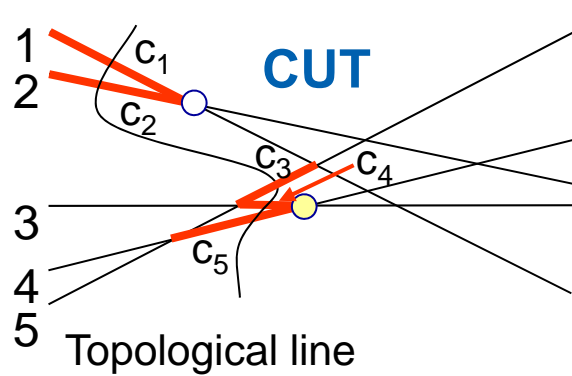
Stack S
Ready vertex first edge idx

c3
c1

Intersect the trees – take the shorter edge



4e) Intersect with neighbors – push into S



Array of line equations E
 $y = a_i x + b$

1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
5	a_5	b_5

UHT array
Delimiting lines indices

1	$-\infty$	2
2	$-\infty$	5
3	5	4
4	5	6
5	3	6
6	$-\infty$	$+\infty$

LHT array
Delimiting lines indices

1	$-\infty$	6
2	$-\infty$	1
3	5	1
4	5	3
5	3	1
6	$+\infty$	$-\infty$

CUT edges N
Pairs of line indices delimiting the edge

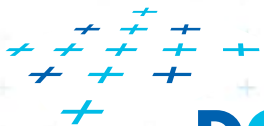
c1	$-\infty$	2
c2	$-\infty$	1
c3	3	1
c4	5	4
c5	5	3

CUT Lines C
Indexes of supporting lines

c1	1
c2	2
c3	5
c4	3
c5	4

Stack S
Ready vertex first edge idx

c4
c1



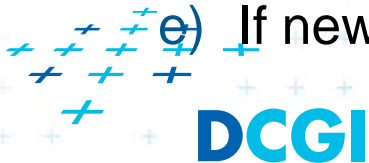
Topological sweep algorithm

TopoSweep(L)

Input: Set of lines L sorted by slope (-90° to 90°), simple, not vertical

Output: All parts of an Arrangement $A(L)$ detected and then destroyed

1. Let C be the **initial (leftmost) cut** – lines in increasing order of slope
2. Create the **initial UHT and LHT** incrementally:
 - a) UHT by inserting lines in decreasing order of slope
 - b) LHT by inserting lines in increasing order of slope
3. By consulting UHT and LHT
 - a) Determine the right endpoints N of all edges of the **initial cut C**
 - b) Store neighboring **lines with common endpoints into stack S**
(ready vertices)
4. Repeat until stack not empty
 - a) Pop next ready vertex from stack S (its upper edge c_i)
 - b) Swap these lines within the cut C ($c_i \leftrightarrow c_{i+1}$)
 - c) Update the horizon trees UHT and LHT
 - d) Consulting UHT and LHT determine new cut edges endpoints N
 - e) If new neighboring edges share an endpoint \rightarrow push them on S



Getting of cut edges from UHT and LHT

- for lines $i = 1$ to n
 - Compare UHT and LHT edges on line i
 - Set the cut lying on edge i to the **shorter edge of these**
- Order of the cuts along the sweep line
 - Order changes at the intersection v only (neighbors)
 - Order of remaining cuts not incident with intersection v does not change
- After changes of the order, test the neighbors for intersections
 - Store intersections right from sweep line into the stack

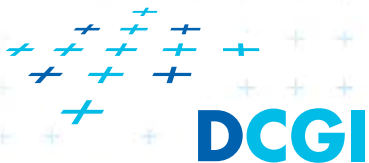


Complexity

- $O(n^2)$ intersections
=> $O(n^2)$ events (elementary steps)
- $O(1)$ amortized time for one step
=> $O(n^2)$ time for the algorithm

Amortized time

= even though a single elementary step can take more than $O(1)$ time, the total time needed to perform $O(n^2)$ elementary steps is $O(n^2)$, hence the average time for each step is $O(1)$.



References

- [Berg] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: Computational Geometry: Algorithms and Applications, Springer-Verlag, 3rd rev. ed. 2008. 386 pages, 370 fig. ISBN: 978-3-540-77973-5, Chapters 8., <http://www.cs.uu.nl/geobook/>
- [Mount] David Mount, - CMSC 754: Computational Geometry, Lecture Notes for Spring 2007, University of Maryland, Lectures 8,15,16,31, and 32. <http://www.cs.umd.edu/class/spring2007/cmsc754/lectures.shtml>
- [Edelsbrunner] Edelsbrunner and Guibas. Topologically sweeping an arrangement. TR 9, 1986, Digital www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-9.pdf
- [Rafalin] E. Rafalin, D. Souvaine, I. Streinu, "Topological Sweep in Degenerate cases", in Proceedings of the 4th international workshop on Algorithm Engineering and Experiments, ALENEX 02, in LNCS 2409, Springer-Verlag, Berlin, Germany, pages 155-156. <http://www.cs.tufts.edu/research/geometry/other/sweep/paper.pdf>





**OPPA European Social Fund
Prague & EU: We invest in your future.**
