Task 3

Gr 5: Mandl, Ankele, Jeran
Gr 12: Pöllitsch, Prünster, Sommer, Vierthaler
Gr 18: Fröhlich, Spitzer, Herrgesell

ECA group C (5, 12, 18)

July 2013

Initial plan

1. GROUP 5: create at all λ - matrices for a specific n, k
   - possible λ - matrices
   - and impossible λ - matrices
   - dead simple
2. GROUP 12: check if given λ-matrices can be realized
   - discard invalid λ-matrices
   - realise valid ones as pointset
   - must be fast
3. GROUP 18: count empty polygons(3,4,5-gons) for given pointsets

checking λ-matrices for validity

core idea: prolog program

- should be easy to implement with simple rules
- should be fast because of logic approach (backtracking & forwardchecking)
- non-deterministic

prolog rules derived from orientation (determinant-calculation):

- `isLeft(p1,p2,px)`
- `isColl(p1,p2,px)`

checking λ-matrices for validity

core idea: prolog program

- should be easy to implement with simple rules
- should be fast because of logic approach (backtracking & forwardchecking)
- non-deterministic

prolog rules derived from orientation (determinant-calculation):

- `isLeft(p1,p2,px)`
- `isColl(p1,p2,px)`

$$\rightarrow \text{FAIL!} \leftarrow$$

problems occured:

performance

- ok for 4 points (< 1 secs)
- not feasible for $n > 4$
- example n $=$ 5:
  - 50 clauses 3 points each $\binom{5}{2} \cdot 5$
  - every point-coordinate (2 coordinates each) can take values from 0..5
    $\rightarrow$ worstcase: $6^{10} \cdot 50 \cdot 12 (= 36.279.705.600)$ operations
- naive approach does not work

ways of improvement

- randomize choice of coordinates $\rightarrow$ no gain
- mathematical approach (screw prolog!) $\rightarrow$ no gain

problems occured:

performance

- ok for 4 points (< 1 secs)
- not feasible for $n > 4$
- example n $=$ 5:
  - 50 clauses 3 points each $\binom{5}{2} \cdot 5$
  - every point-coordinate (2 coordinates each) can take values from 0..5
    $\rightarrow$ worstcase: $6^{10} \cdot 50 \cdot 12 (= 36.279.705.600)$ operations
- naive approach does not work

ways of improvement

- randomize choice of coordinates $\rightarrow$ no gain
- mathematical approach (screw prolog!) $\rightarrow$ no gain

$$\rightarrow \text{FAIL!} \leftarrow$$

no problem because ...

group 5 can generate all possible pointsets and check for duplicates
$\rightarrow$ new plan

New plan

1. GROUP 5: create all pointsets for a specific n, k
   - only pointsets with valid λ - matrices
2. GROUP 12: parse output from group 5 and transform to correct input for group 18
3. GROUP 18: count empty polygons(3,4,5-gons) for given pointsets

create λ - matrices

K = Number of points
N = Gridsize
Procedure to create all λ - matrices for a given point set

- Create all possible pointsets for a fix K and N
- For each pointset calculate minimal λ matrix (Task 2 Matlab script)
- Save all unique minimal λ matrices and one pointset
- Store the resulting unique pointsets in file for further processing

Fingerprint(min. λ matrix)

Changes to cope with collinear points
Get convex hull for given pointset
For each point on the hull

- Calculate difference vector to all other points
- Get angle between these vectors and first difference vector on convex hull
- Sort points in ascending order according to the distance
- Sort points again according to the angle (clockwise)
- Calculate λ matrix for this labeling

Filter collinear and non-collinear point sets

Collinearity can be checked during calculation of λ matrix.
For each point pair:

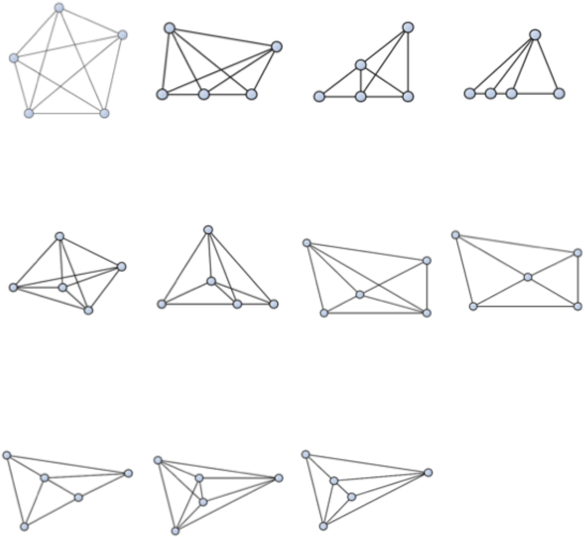
- Create triangle with every other point
- Check if point is left or right using the determinant
- If determinant is zero the three points lie on the same line
- Count points on left and right side
- Set entries in the λ matrix

Return λ matrix and number of collinear points.
Take point sets with/without collinear points or both.

Overview group 18

- Classification, $n \leq 5$
- Statistical analysis of classification on a 5×5 grid
- Algorithm for counting empty polygons
- Idea!
- executive summary

Classification $n = 5$



5 point sets in a 5×5 grid

class 5 (5-convex)	11628	21,89 %
class 6 (5-convex, 1x collconvex)	13668	25,73 %
class 7 (5-convex, 2x collconvex)	1436	2,7 %
class 8 (5-convex, >2x collconvex)	1284	2,42 %
class 9 (4-convex)	12800	24,1 %
class 10 (4-convex, 1xcollconvex)	2336	4,4 %
class 11 (4-convex, 1xcollinear)	6420	12,09 %
class 12 (4-convex, 2xcollinear)	578	1,09 %
class 13 (4-convex, 1xcollinear, 1x collconvex)	1060	2 %
class 14 (3-convex)	624	1,17 %
class 15 (3-convex, 1xcollconvex)	1284	2,42 %

Algorithm for counting

- first approach: for triangles brute force
- tetragons: recursive: deleting one point, inside-out test, count tetragon in $n-1$-gon
- problem: what if point is a collinear middlepoint?

- Idea: our classification leads to number of polygons without calculating them

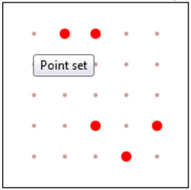
Webinterface

- PHP-Webservice for other groups
- input: $n \leq 5$ Pointset on a 5×5 Grid
- output: classification and number of n-gons
- `http://www.upinthecloud.at/eca/getemptypolygons.php?pointset=0_0_2_1_4_0_2_2_2_4` ▸ Link

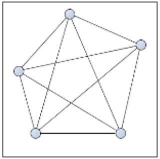
Webdemo for 5 $\times$ 5 Grid

ECA task 3

POINT SETS ON AN INTEGER GRID (5X5)



POINT SET DETAILS



Classification

Generate random 5-point set in a 5x5 grid

Point set details:

Classification: 5
Empty triangles: 10
Empty convex, not collinear triangles: 10
Empty tetragons: 5
Empty convex, not collinear tetragons: 5
Empty convex and collinear tetragons: 0
Empty not convex, not collinear tetragons: 0
Empty not convex, but collinear tetragons: 0
Empty pentagons: 1
Empty convex, not collinear pentagons: 1
Empty convex and collinear pentagons: 0
Empty not convex, not collinear pentagons: 0
Empty not convex, but collinear pentagons: 0

- `http://www.upinthecloud.at/eca/ecatask3.html` Link

executive summary - group 18

- Java tool for generating all pointsets for arbitrary grid size
- Java classification tool for classifying and counting n-gons for a given pointset (arbitrary grid size)
- JUnit tests for proof of concept
- PHP-Webservice for classifying and counting n-gons for a given 5-pointset (5×5 grid)
- Demo website for 5-pointsets on a 5×5 grid

thank you!