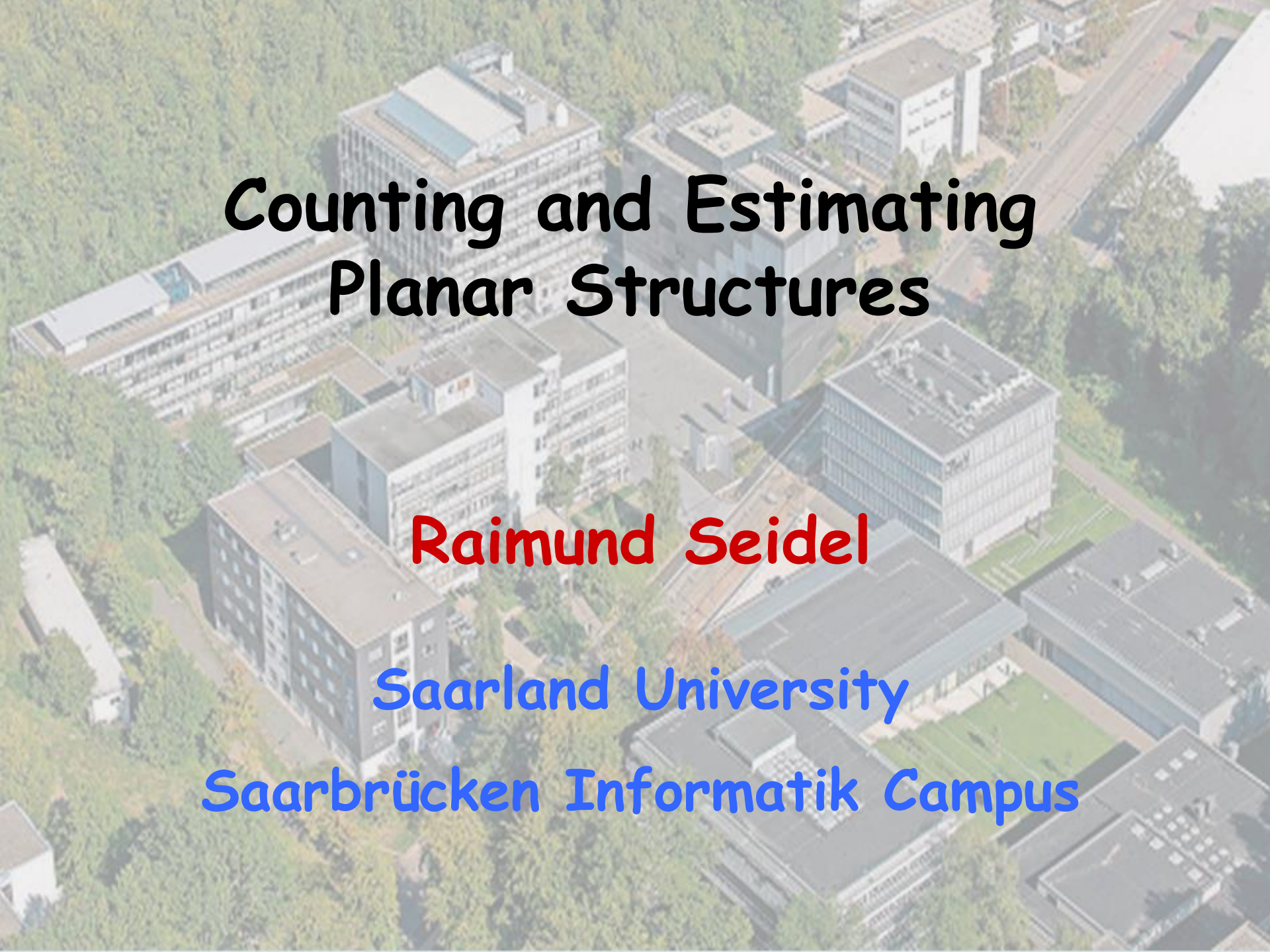
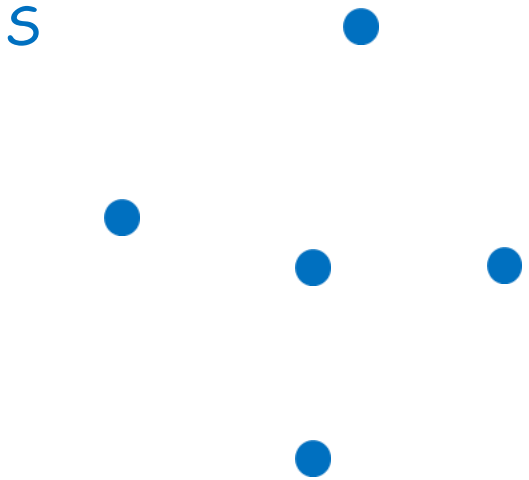


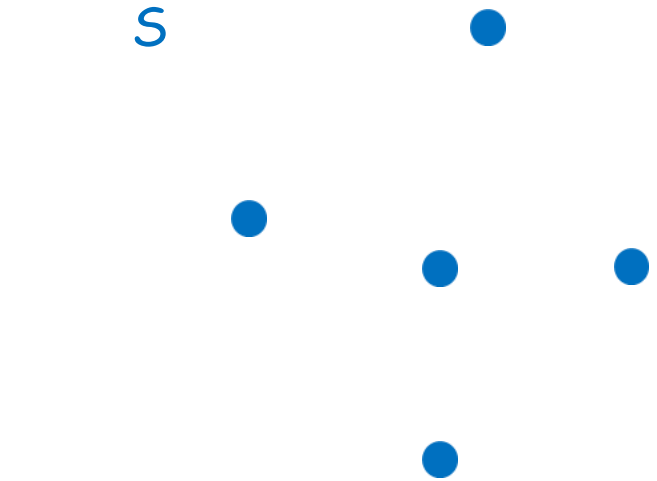
Counting and Estimating Planar Structures

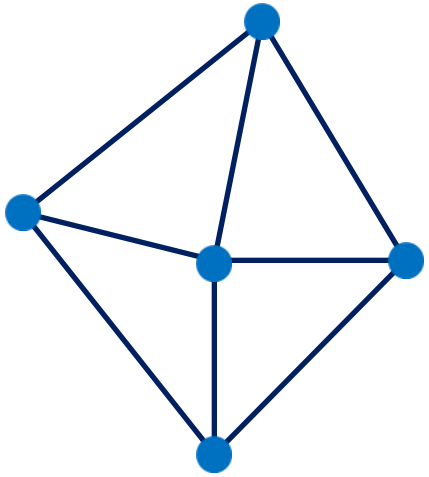
Raimund Seidel

Saarland University

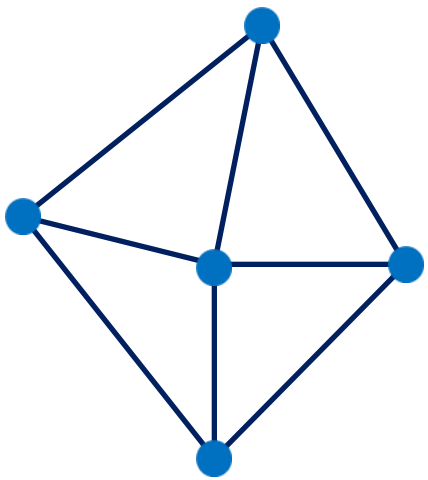
Saarbrücken Informatik Campus

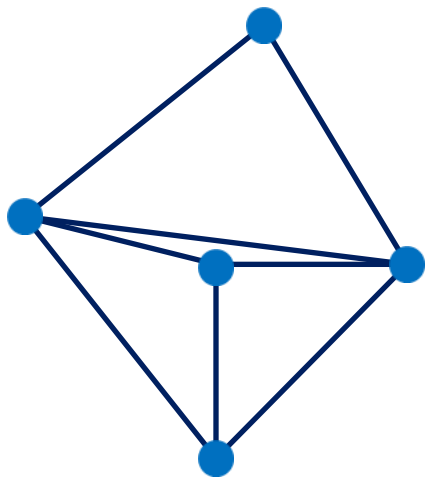

S


S

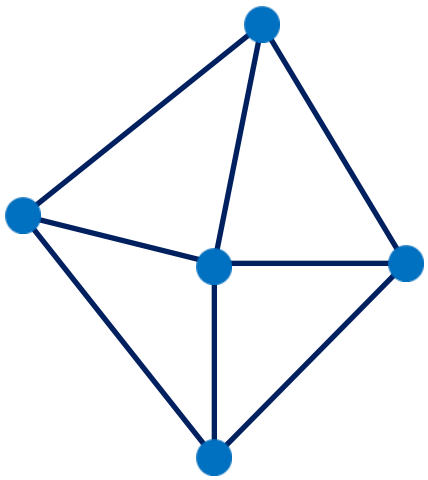


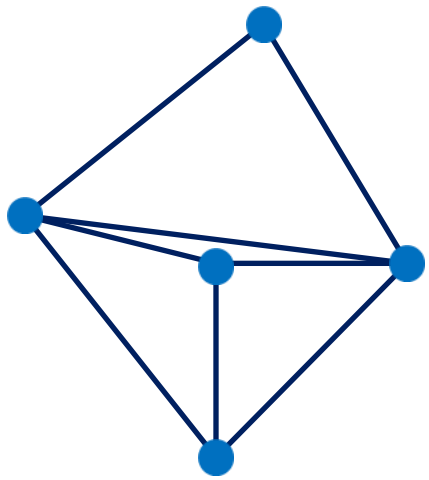
S

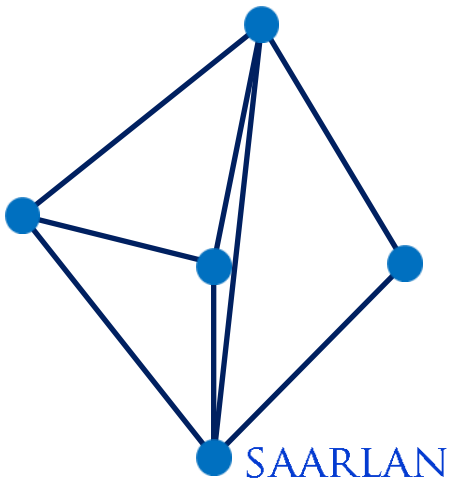




S



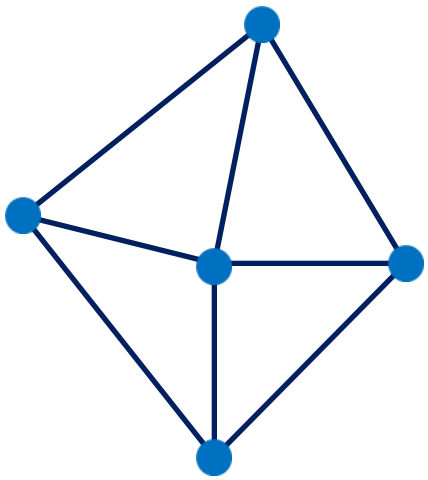


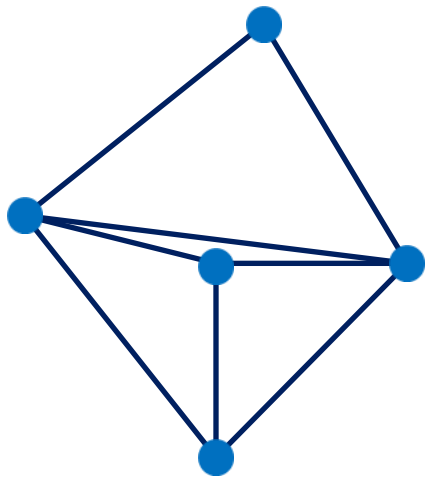


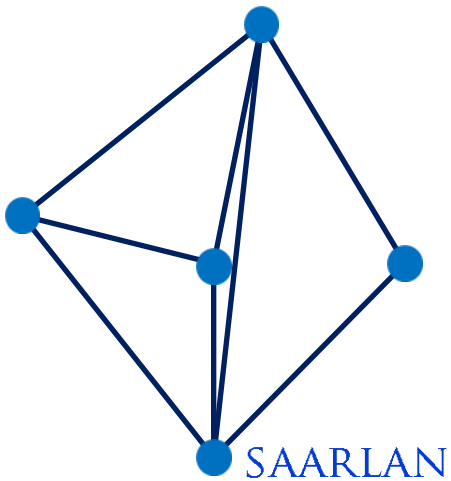
S

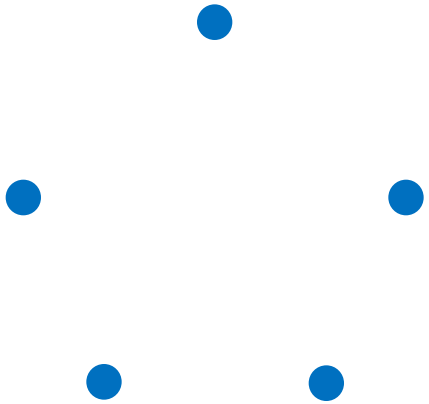
$tr(S)$ = # triangulations of S

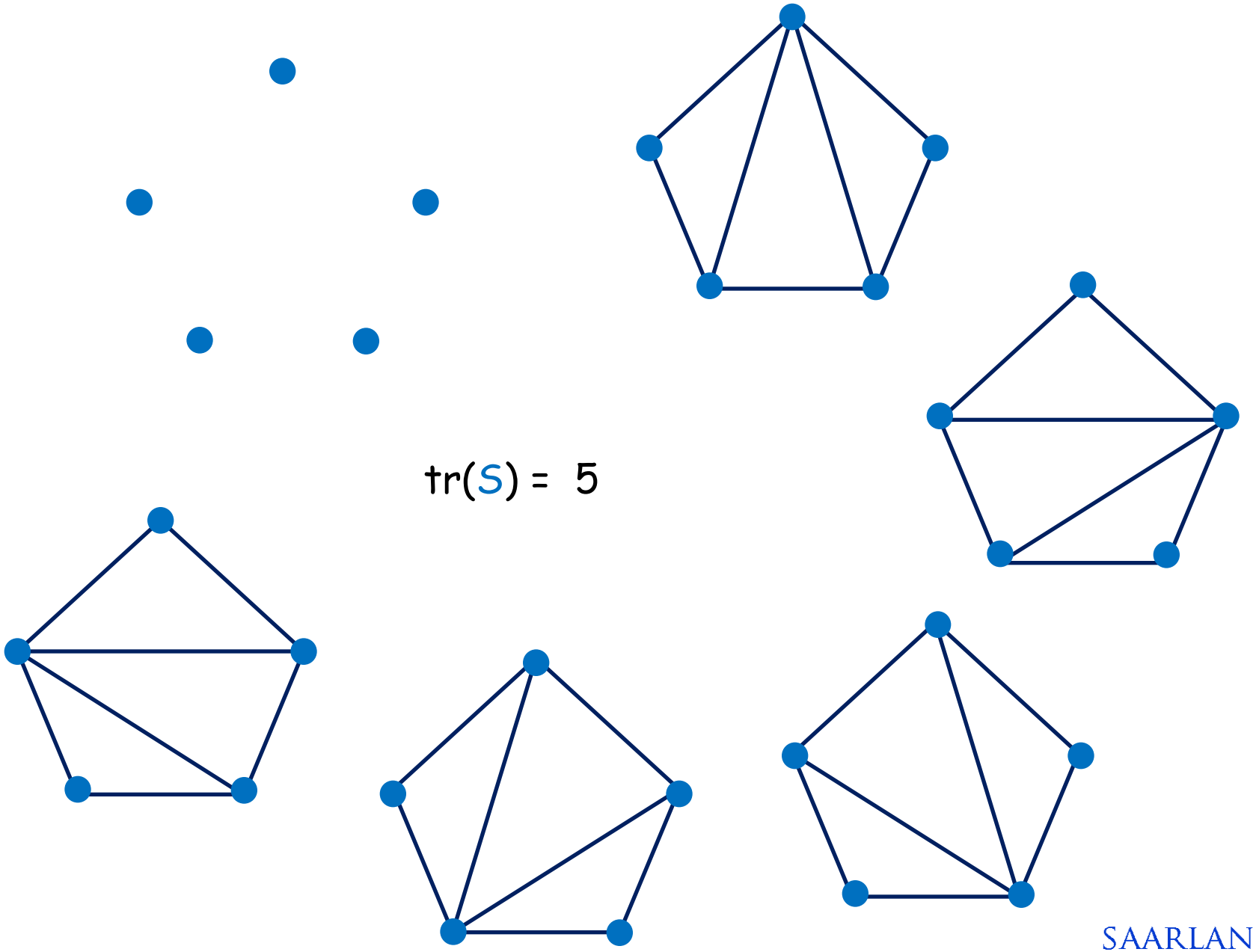
$tr(S) = 3$



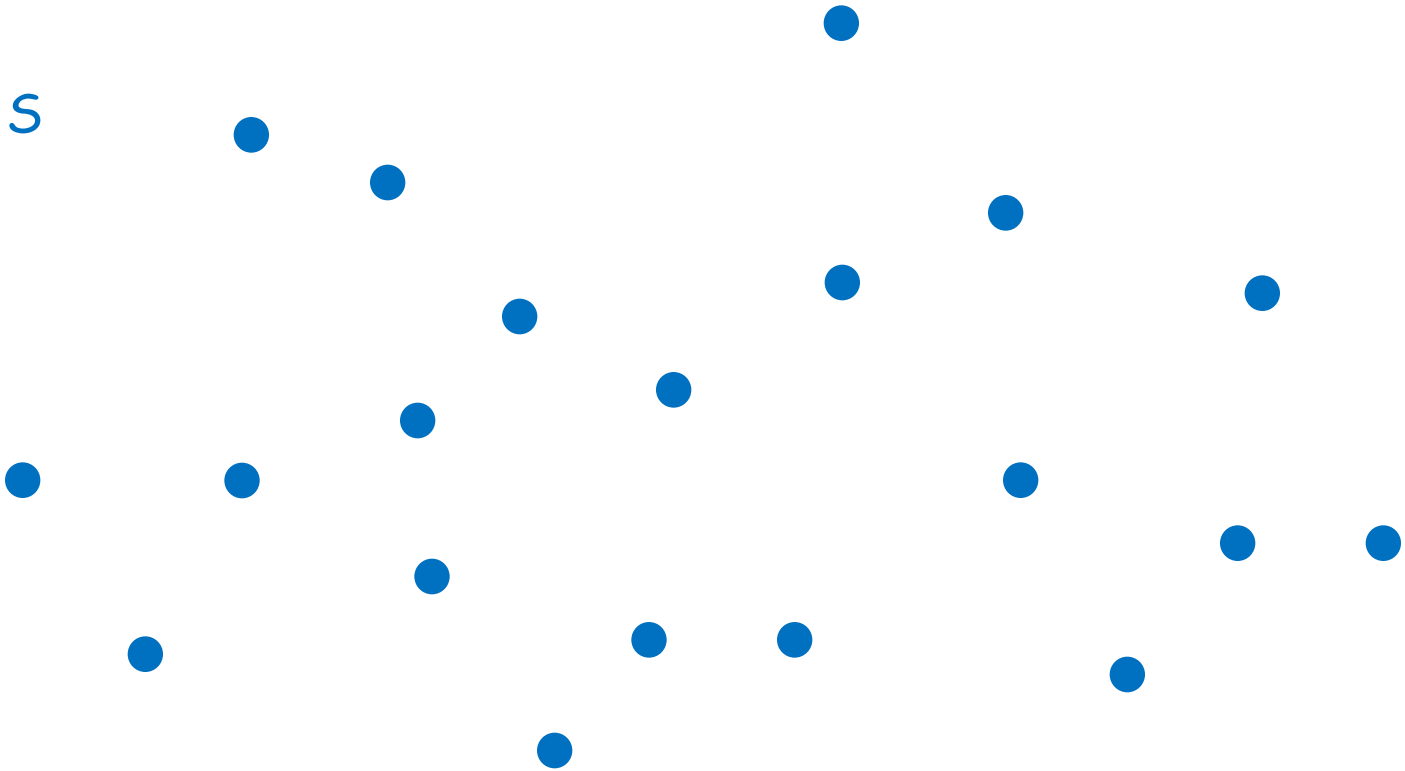








tr(S) = ?



$tr(S)$ = # triangulations of S

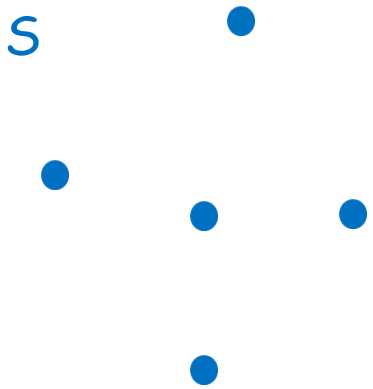
$Tr(n) = \max\{ tr(S) : |S|=n \}$

$tr(n) = \min\{ tr(S) : |S|=n \}$

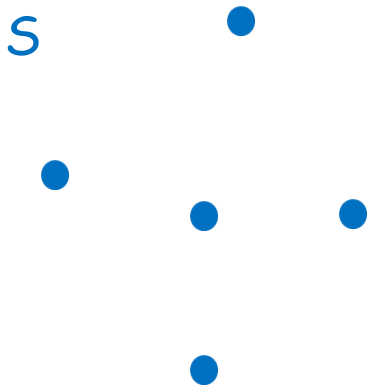
Questions:

- Bounds for $Tr(n)$
- Bounds for $tr(n)$
- Computing $tr(S)$
- Approximating/Estimating $tr(S)$
- Enumerating all triangulations of S
- random sampling a triangulation of S

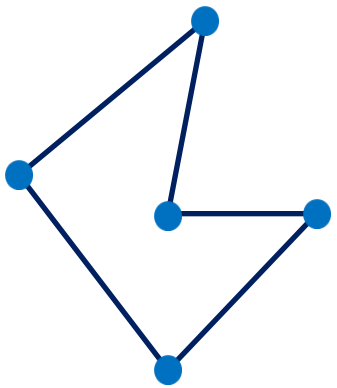
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



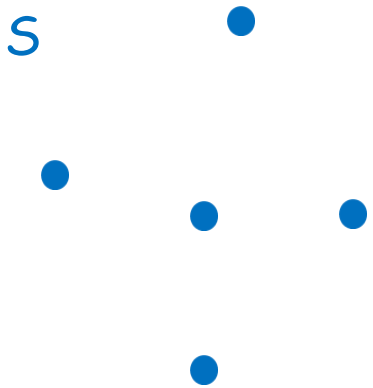
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



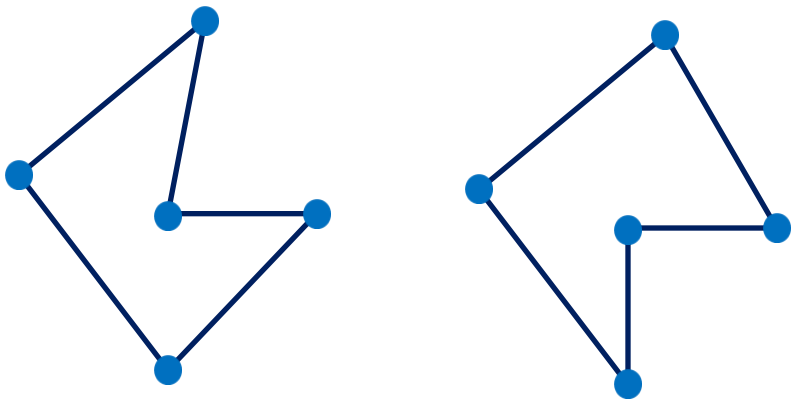
hc(S) = ?



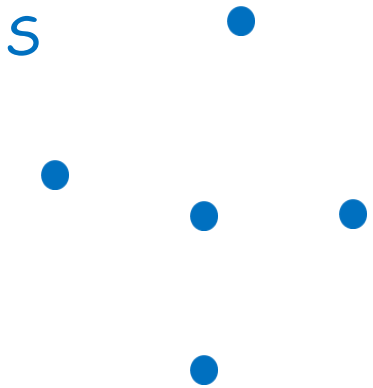
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



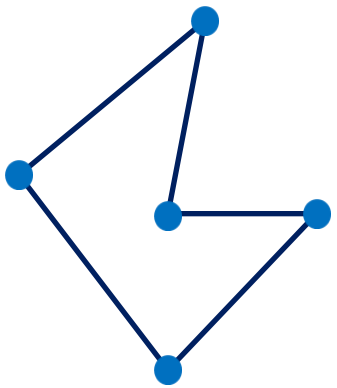
hc(S) = ?

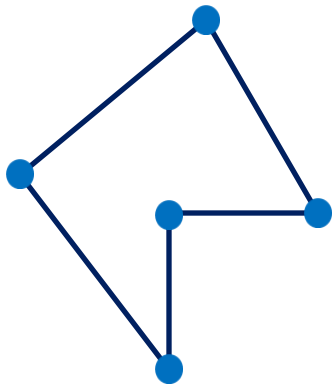


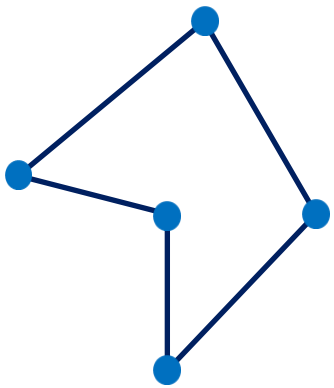
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



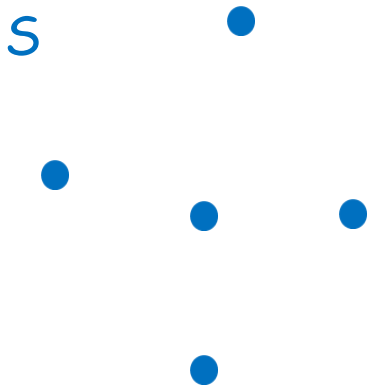
hc(S) = ?



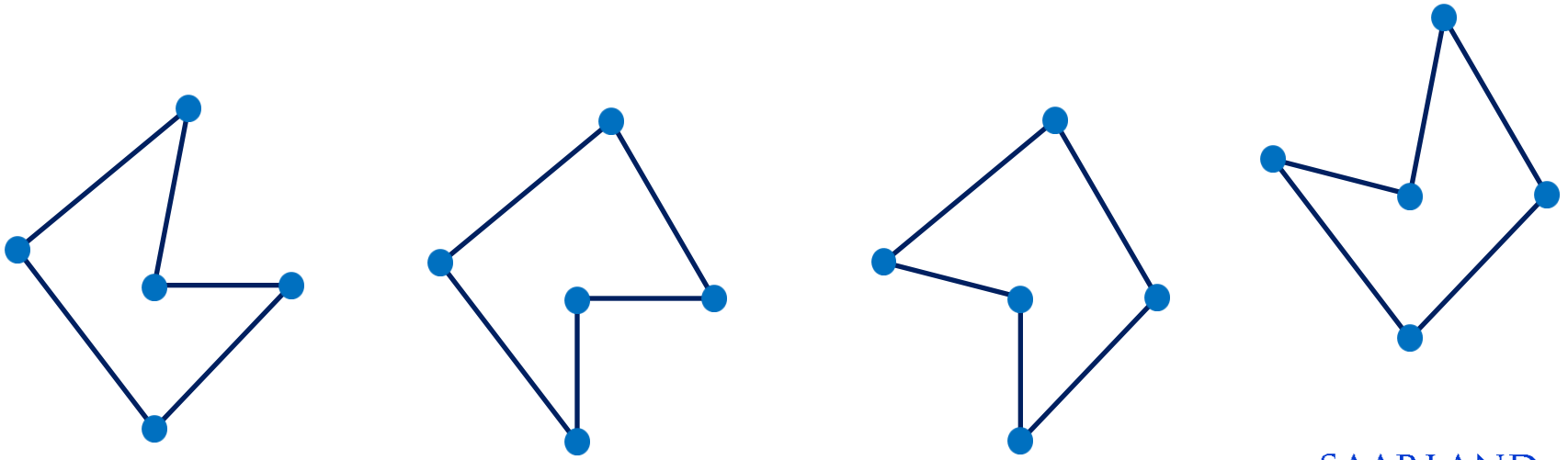




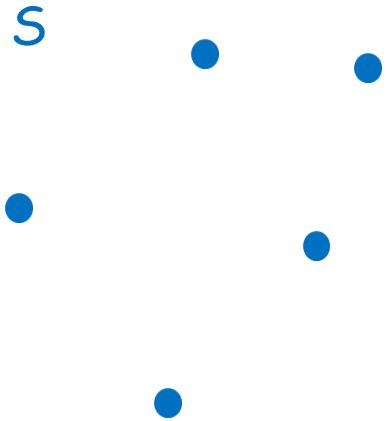
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



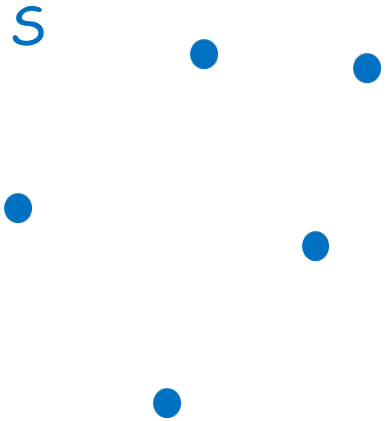
hc(S) = 4



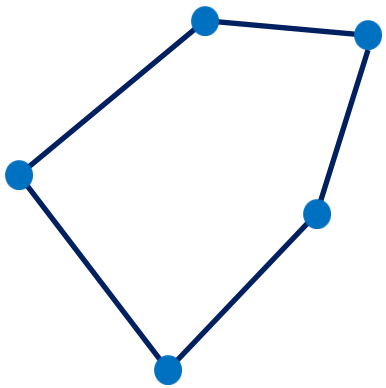
hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



hc(S) = # **non-crossing** straight-edge hamiltonian cycles on S



hc(S) = 1



hc(S) = # non-crossing straight-edge hamiltonian cycles on S

„plane“

hc(S) = # plane hamiltonian cycles on S

$Hc(n) = \max\{ hc(S) : |S|=n \}$

$hc(n) = \min\{ hc(S) : |S|=n \}$

$hc(S)$ = # plane hamiltonian cycles on S

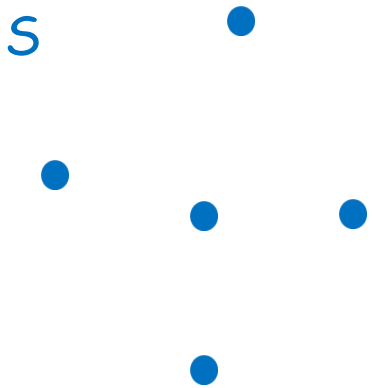
$Hc(n) = \max\{ hc(S) : |S|=n \}$

$hc(n) = \min\{ hc(S) : |S|=n \}$

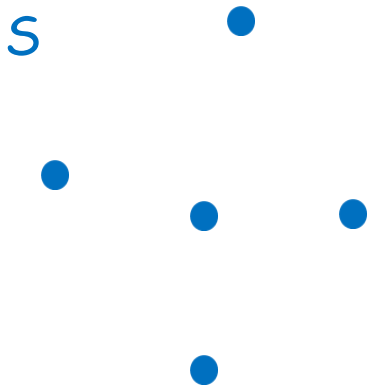
Questions:

- Bounds for $Hc(n)$
- Bounds for $hc(n)$
- Computing $hc(S)$
- Approximating/Estimating $hc(S)$
- Enumerating plane hamiltonian cycles on S
- random sampling a plane hamiltonian cycle on S

pm(S) = # plane maximum matchings on S

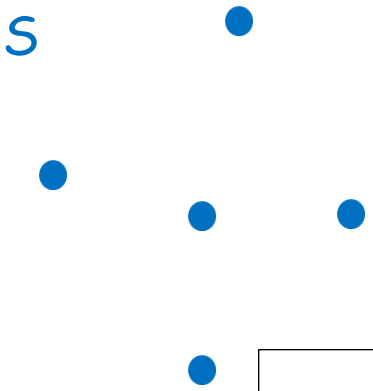


pm(S) = # plane maximum matchings on S

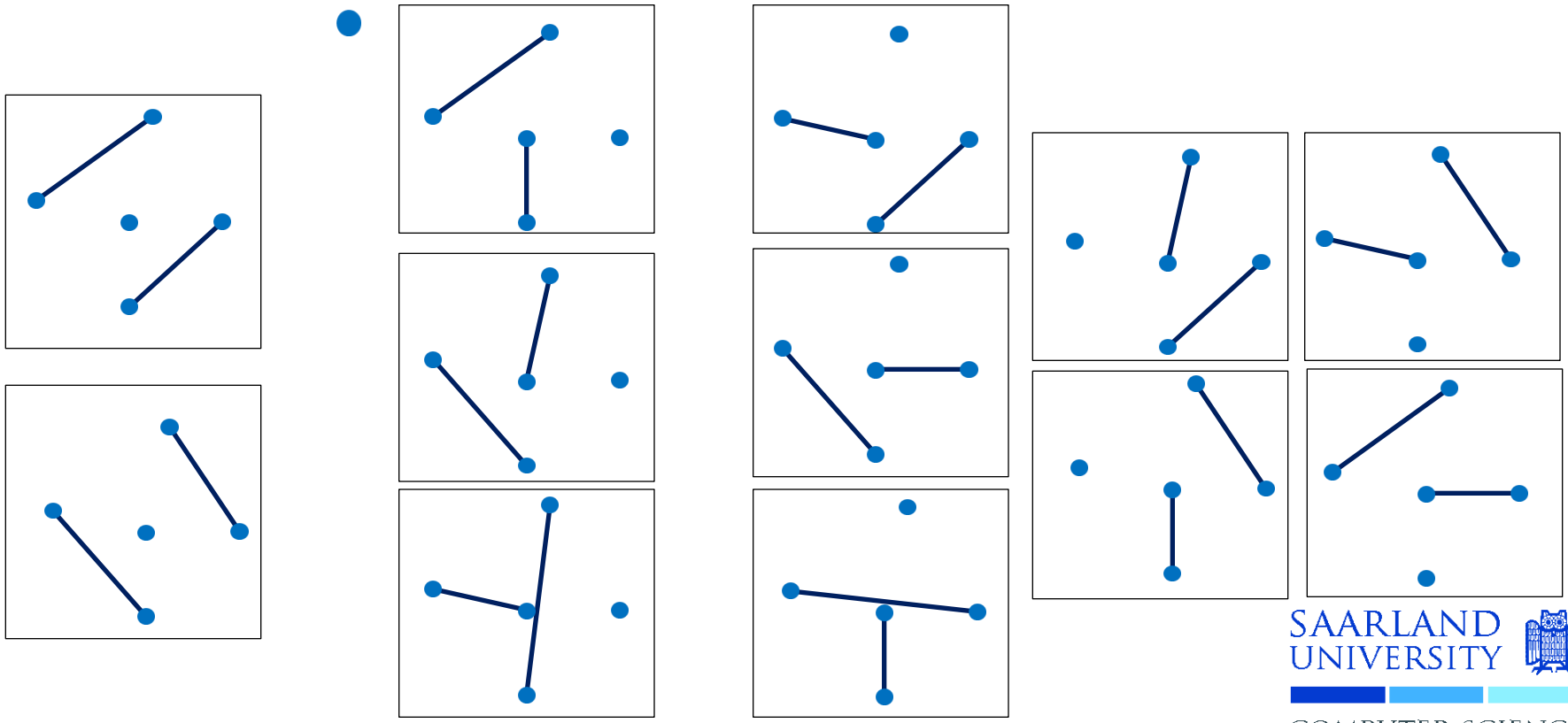


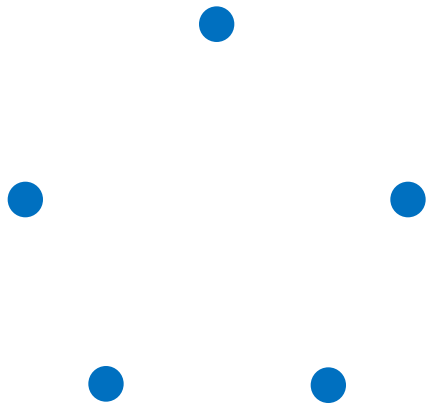
pm(S) = 12

pm(S) = # plane maximum matchings on S

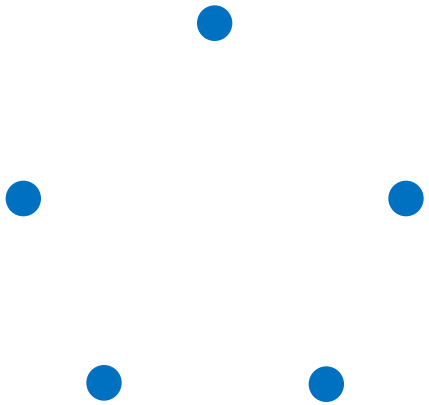


pm(S) = 12

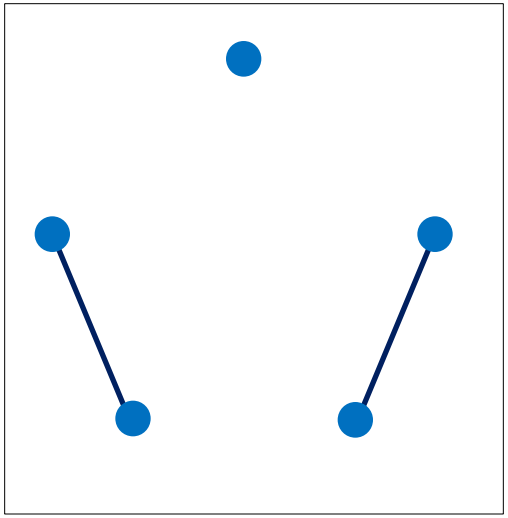


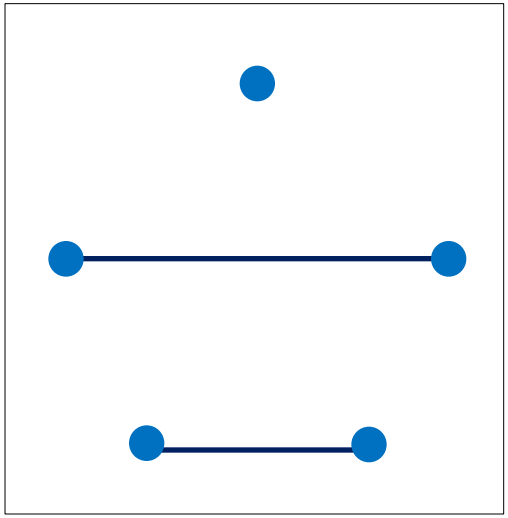


$pm(S) = ?$



pm(S) = 10





$pm(S)$ = # plane maximum matchings on S

$$Pm(n) = \max\{ pm(S) : |S|=n \}$$

$$pm(n) = \min\{ pm(S) : |S|=n \}$$

Questions:

- Bounds for $Pm(n)$
- Bounds for $pm(n)$
- Computing $pm(S)$
- Approximating/Estimating $pm(S)$
- Enumerating plane maximum matchings on S
- random sampling a plane maximum matching on S

Other classes of "plane structures" on S:

- plane spanning trees
- plane cycle covers
- plane graphs
- plane convex covers
- ...

$xx(S)$ = # plane xxxxx on S

$Xx(n) = \max\{ xx(S) : |S|=n \}$

$xx(n) = \min\{ xx(S) : |S|=n \}$

Questions:

- Bounds for $Xx(n)$
- Bounds for $xx(n)$
- Computing $xx(S)$
- Approximating/Estimating $xx(S)$
- Enumerating plane xxxxx on S
- random sampling a plane xxxxx on S

Inspite of decades of research each problem is either

trivial

or

not answered satisfactorily.

Triangulations

tr(S) = # triangulations of S

Tr(n) = max{ tr(S) : $|S|=n$ }

tr(n) = min{ tr(S) : $|S|=n$ }

Questions:

- Bounds for Tr(n)
- Bounds for tr(n)
- Computing tr(S)
- Approximating/Estimating tr(S)
- Enumerating all triangulations of S
- random sampling a triangulation of S

$\Omega(2.43^n) \leq \mathrm{tr}(S) \leq O(30^n)$ $\quad\quad$ $n=|S|$

Aichholzer, Hackl, Vogtenhuber, Huemer, Hurtado, Krasser, McCabe, Seidel, Sharir, Sheffer, Noy, Flajolet, …
Ajtai, Chvátal, Newborn, Szemerédi, Smith, Seidel, Santos, Sharir, Welzl, Sheffer, Dumitrescu, Schulz, Toth, …

$$\Omega(8.65^n) \leq \mathrm{Tr}(n) \leq O(30^n)$$

$$\Omega(2.43^n) \leq \mathrm{tr}(n) \leq O(3.464^n)$$

McCabe, Seidel, EuroCG2003

tr(h,m) = min{ tr(S) | S has h extreme points and m "interior" points }

Theorem 1:

$tr(h,m) \geq \lambda \cdot (30/11)^h \cdot (11/5)^m$

$\geq \lambda \cdot 2.7272^h \cdot 2.2^m$ $\qquad \lambda \geq 0.0414$

Theorem 2: For every fixed h:

$tr(h,m) \geq \Omega(2.63^m)$

Computing tr(S)

enumeration via reverse search $\Omega(\mathrm{tr}(S)) \geq \Omega(2.43^n)$
(Avis, Fukuda, Bespamyatnikh)

divide-and-conquer using „triangulation paths" ???
(Aichholzer)

dynamic programming ???
(Ray, Seidel)

sweep using „triangulation paths" ???
(Alvarez, Bringmann, Ray)

dynamic programming using „onion structure" $O(3.1414^n)$
(Alvarez, Bringmann, Curticapean, Ray)

via reduction to path counting in DAGs $O(2^n n^2)$

(Alvarez, S.)

Today: $O(2^n n^2)$ algorithm

via reduction to path counting in DAGs

"aggregative" …. substantially faster than enumeration

$\mathrm{tr}(S) \geq \Omega(2.43^n)$

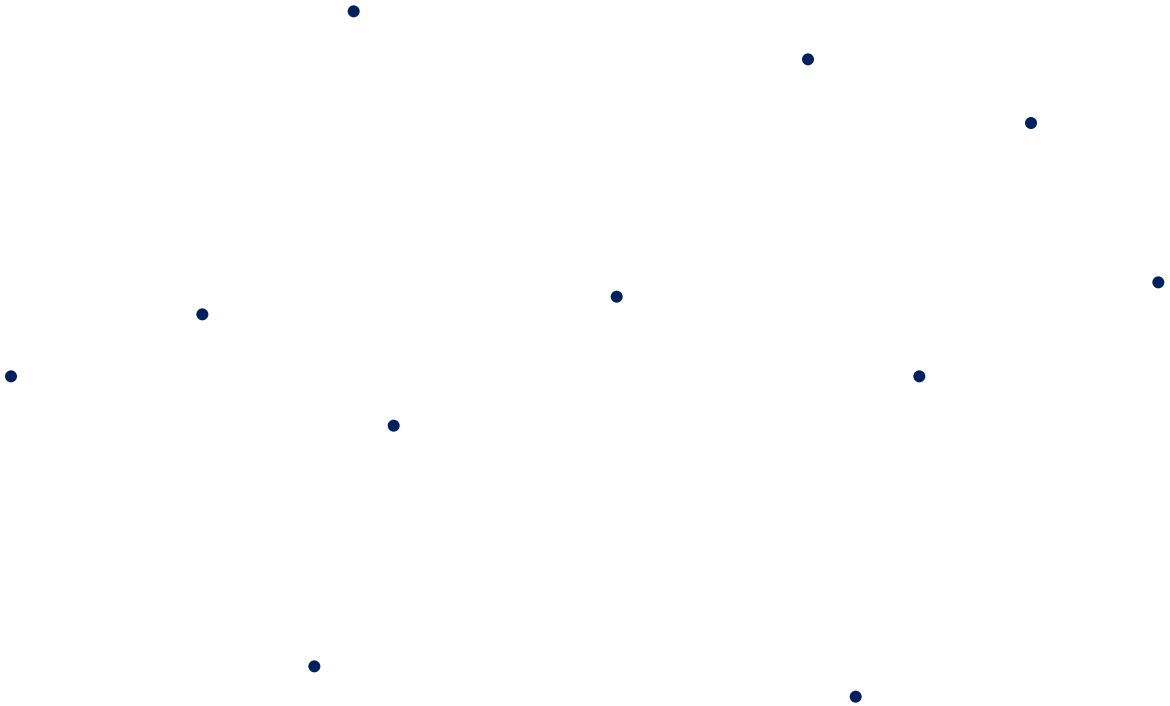
Today: $O(2^n n^2)$ algorithm

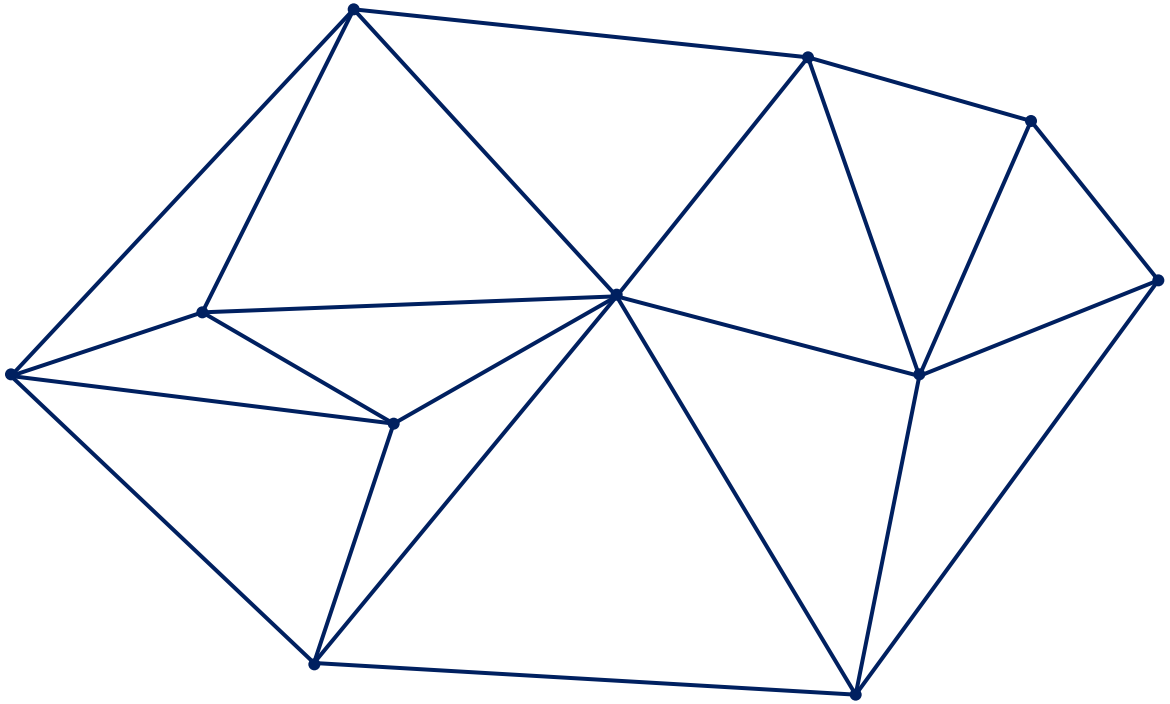
via reduction to path counting in DAGs

"aggregative" …. substantially faster than enumeration

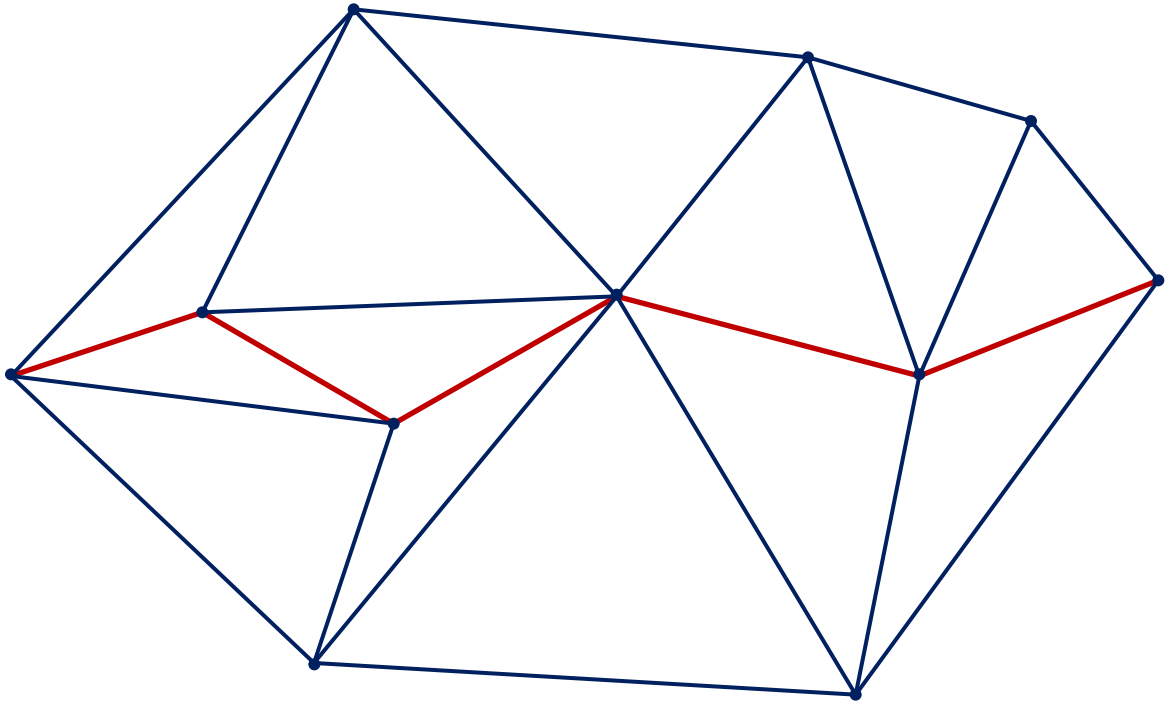
$\mathrm{tr}(S) \geq \Omega(2.43^n)$

Exponential space usage: $O(2^n n)$



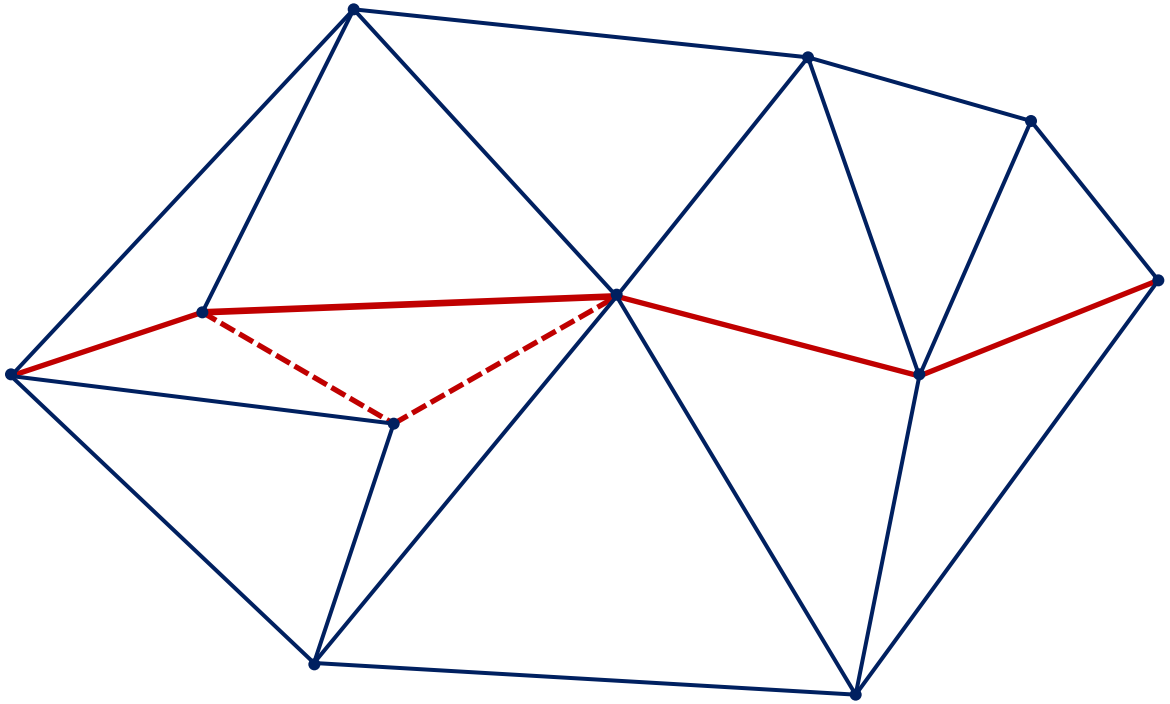


monotone chain



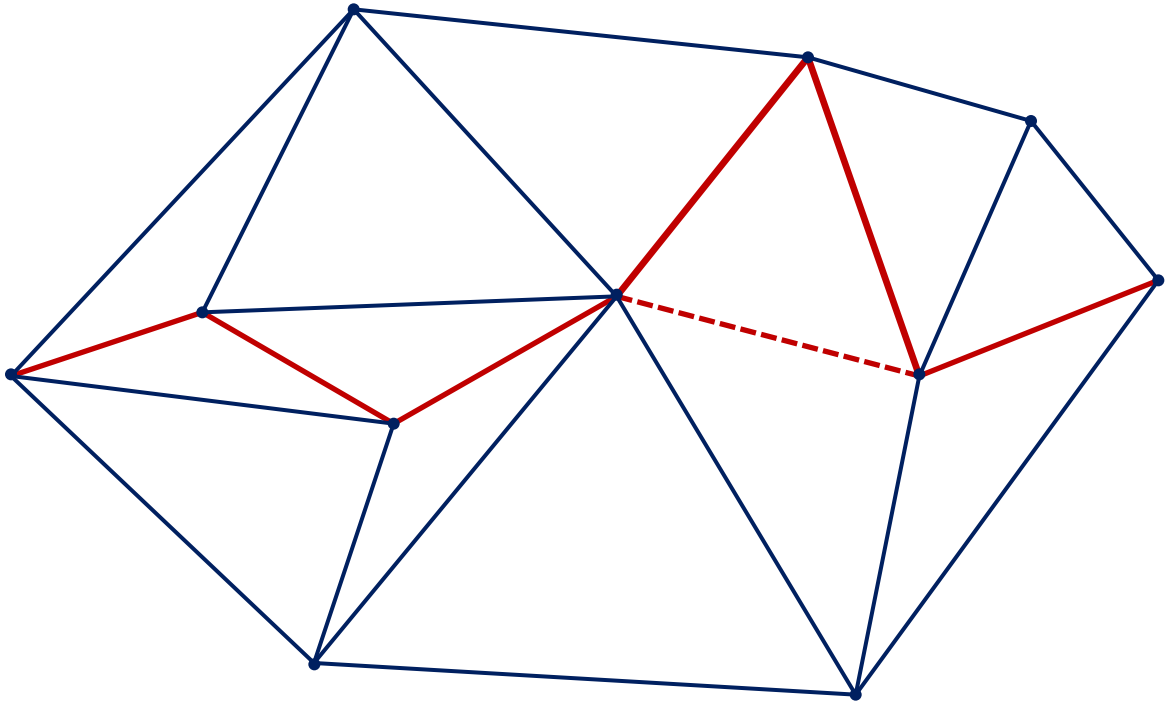
advance

(losing a vertex)



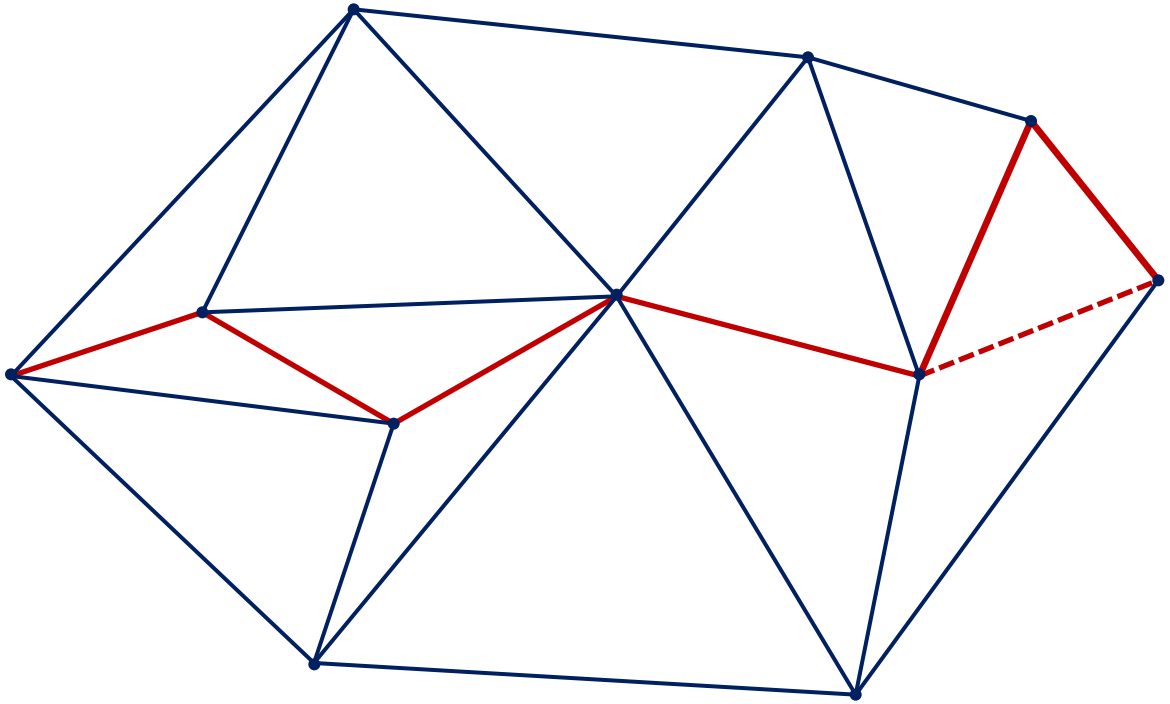
advance

(winning a vertex)



advance

(winning a vertex)



Claim:
For every monotone chain C in a triangulation there is an advance.

Claim:
For every monotone chain C in a triangulation there is an advance,

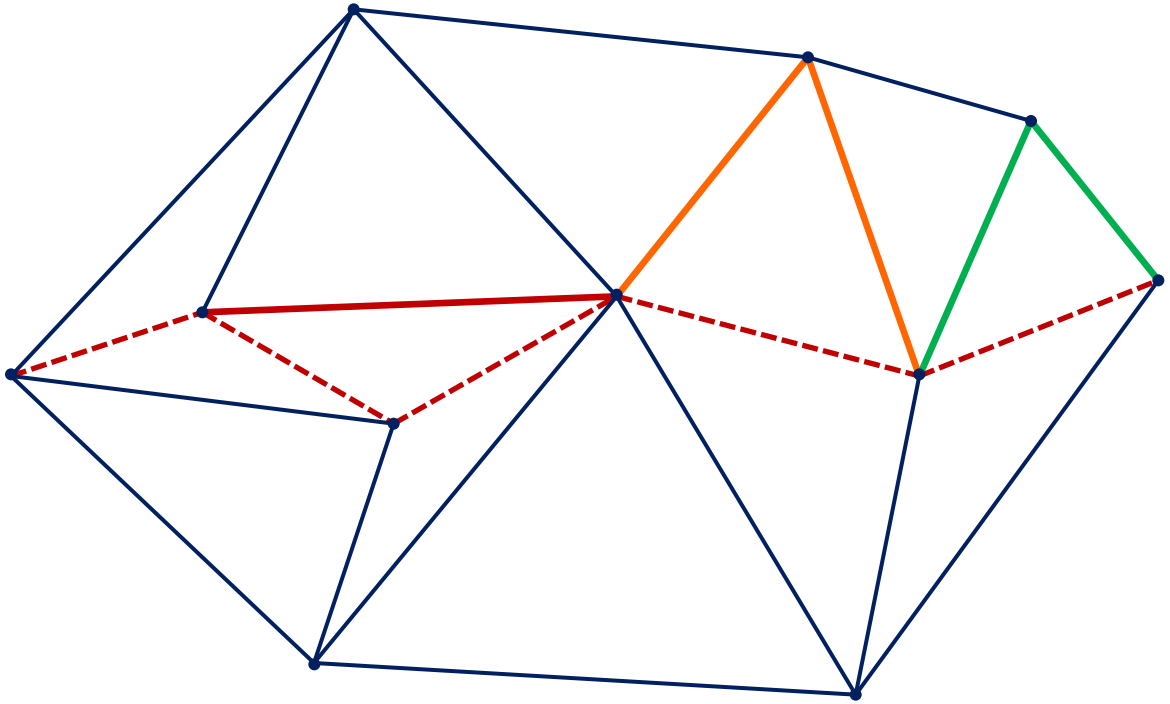
unless C is the chain formed by the upper hull.

Claim:
For every monotone chain C in a triangulation there is an advance,

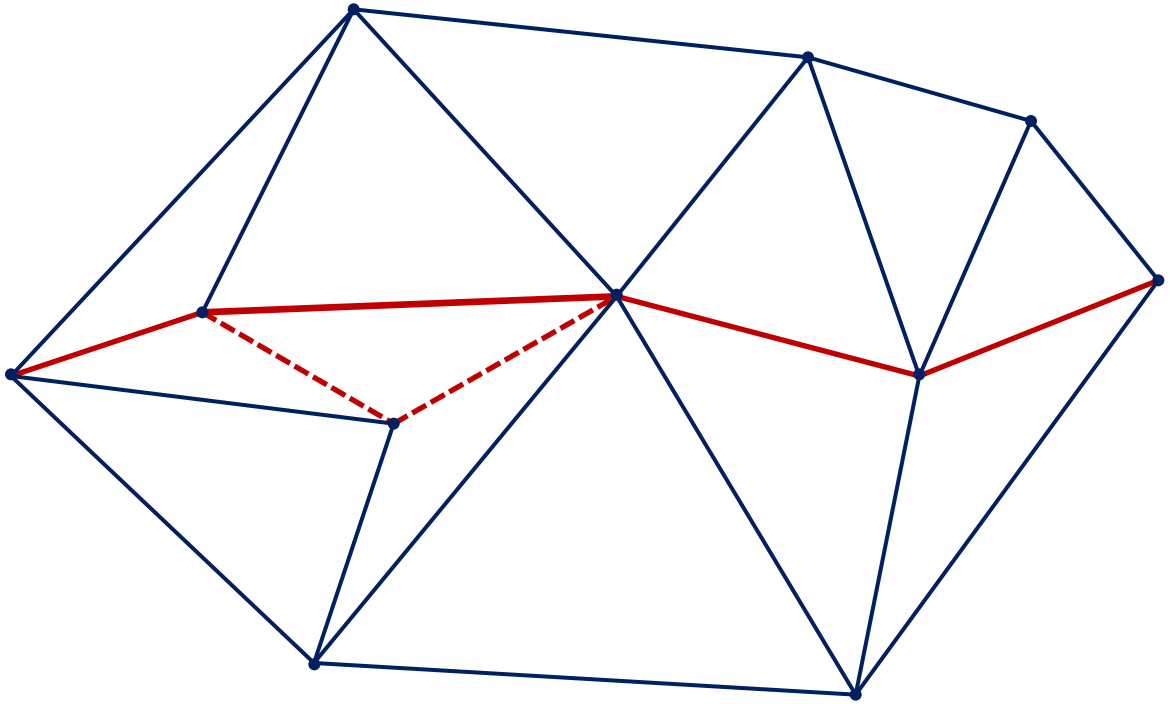
unless C is the chain formed by the upper hull.

Corollary:
For every monotone chain C in a triangulation there is a **leftmost** advance,

unless C is the chain formed by the upper hull.



leftmost advance

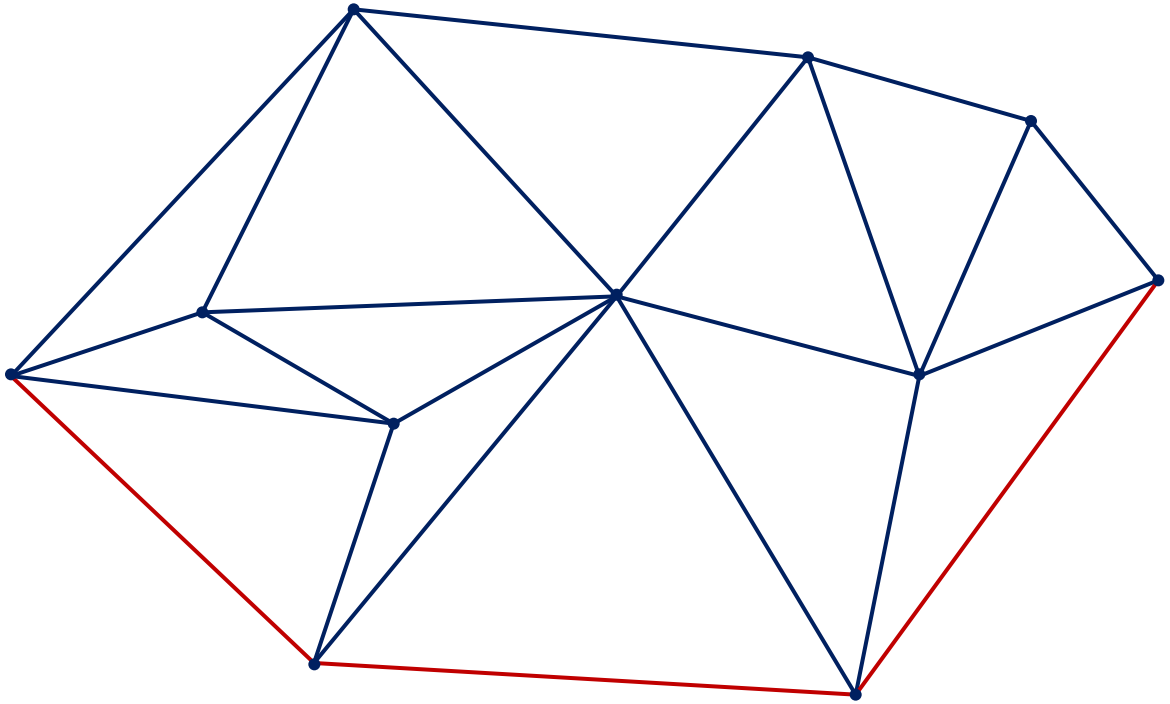


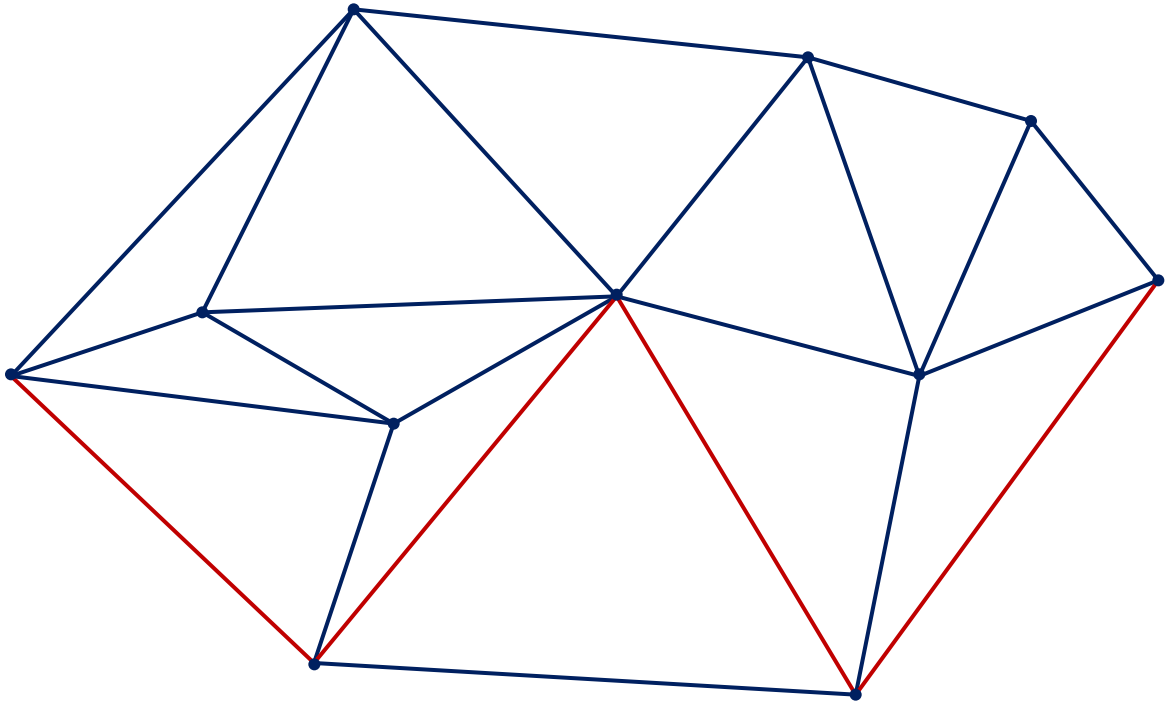
Leftmost advancing monotone chain sweep of a triangulation T

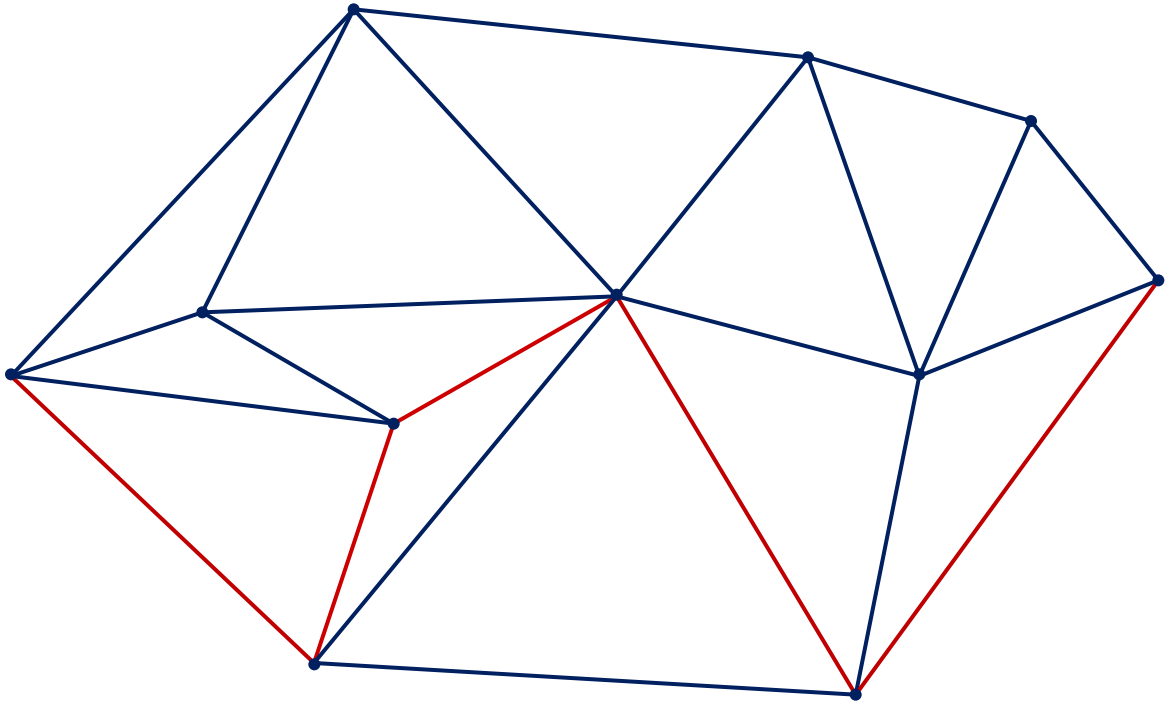
Sequence $C_0, C_1, \ldots, C_t$ of monotone chains of T such that

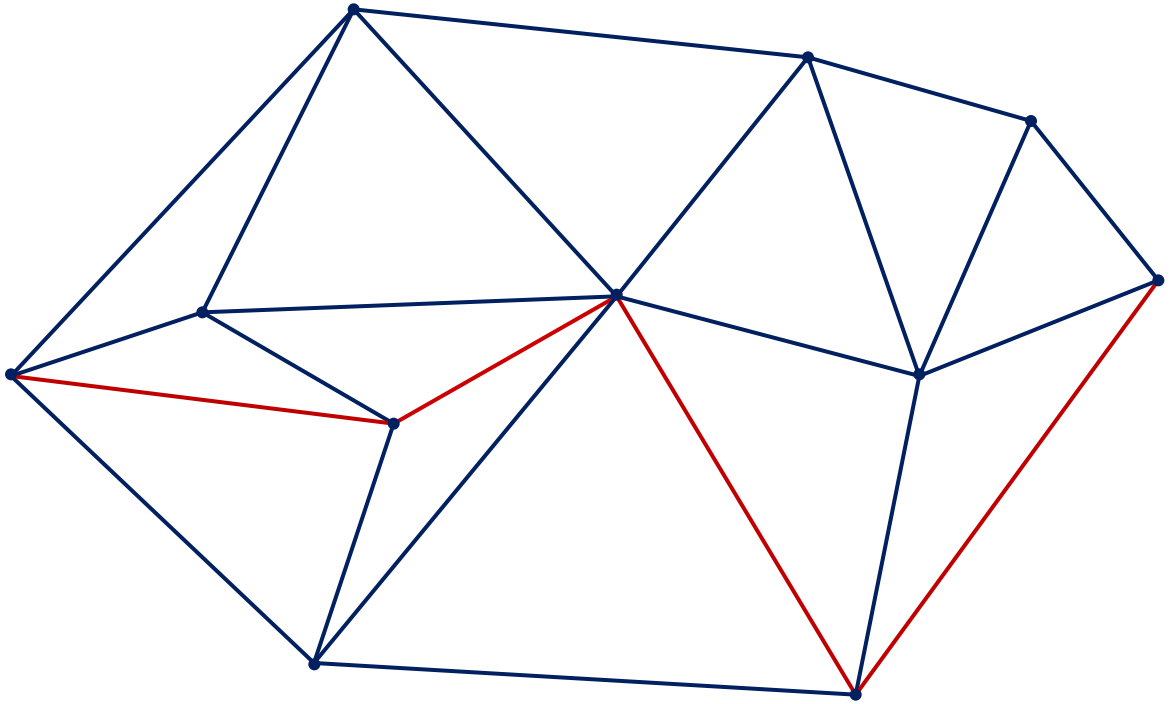
C_0 is the lower convex hull
C_t is the upper convex hull
C_{i+1} obtained from C_i by a leftmost advance

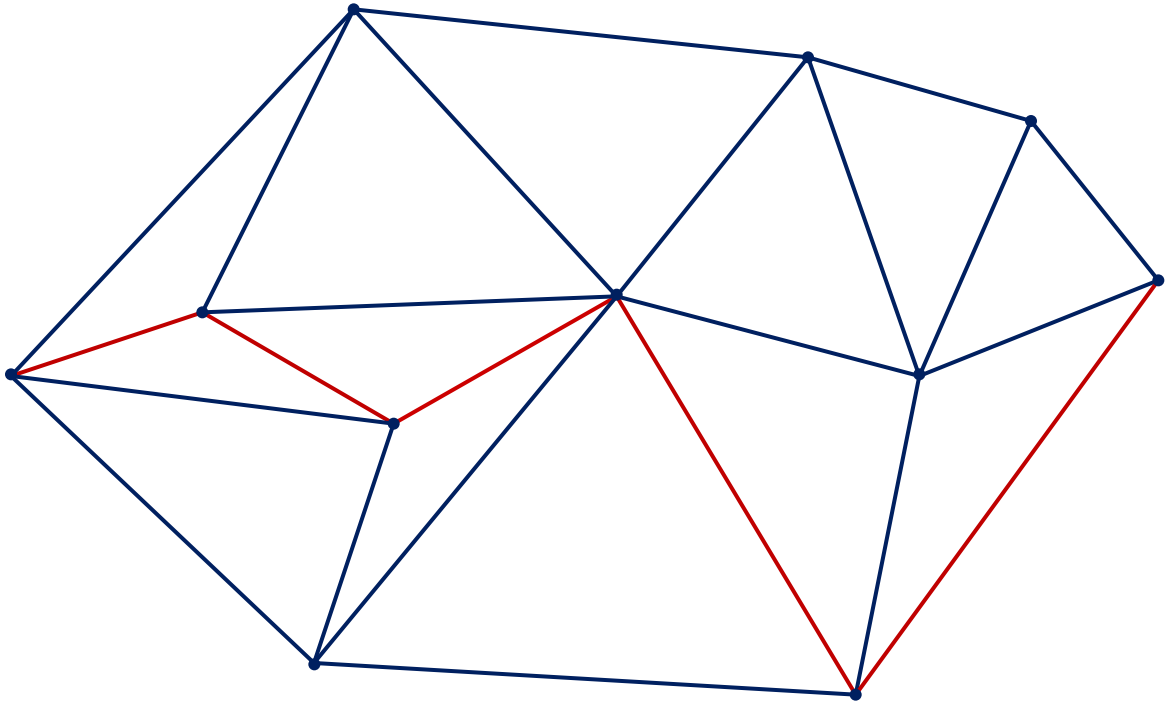
Claim:
Every triangulation has a unique leftmost advancing sweep.

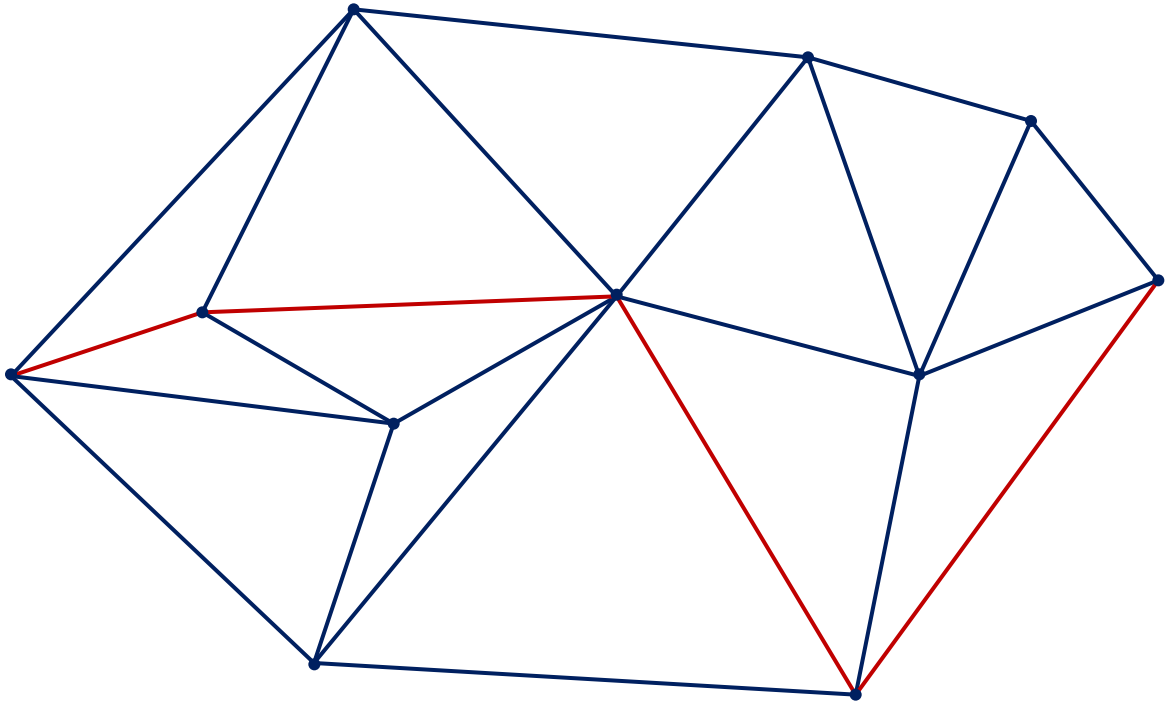


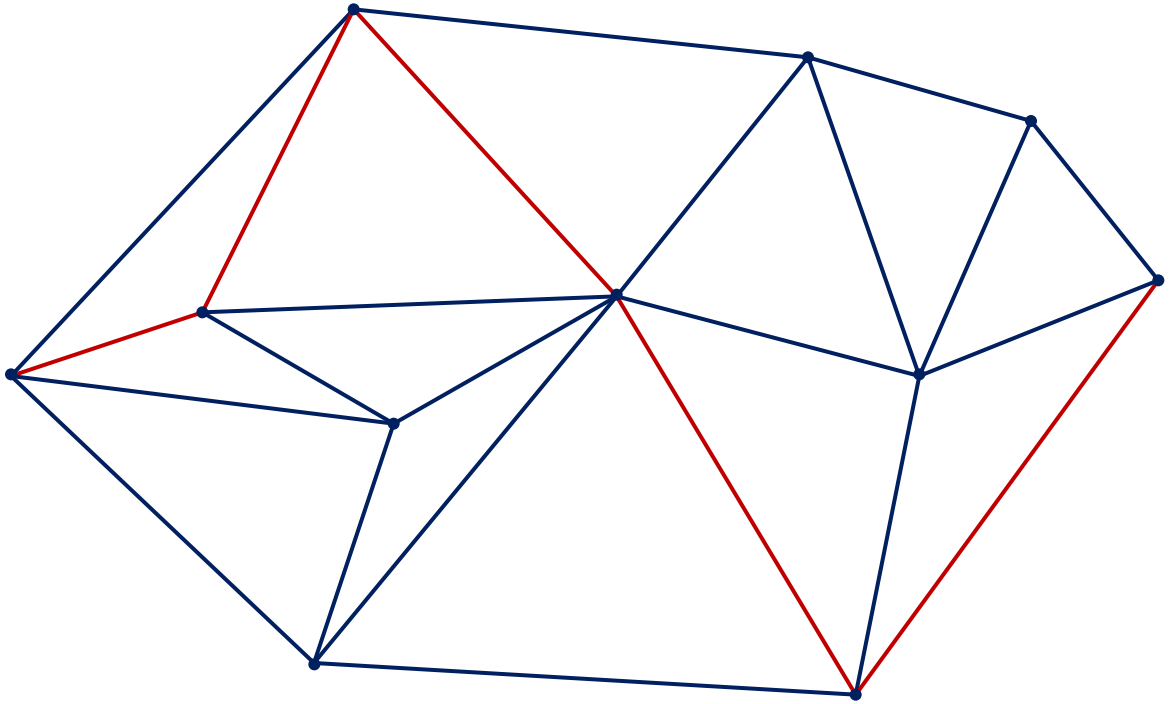


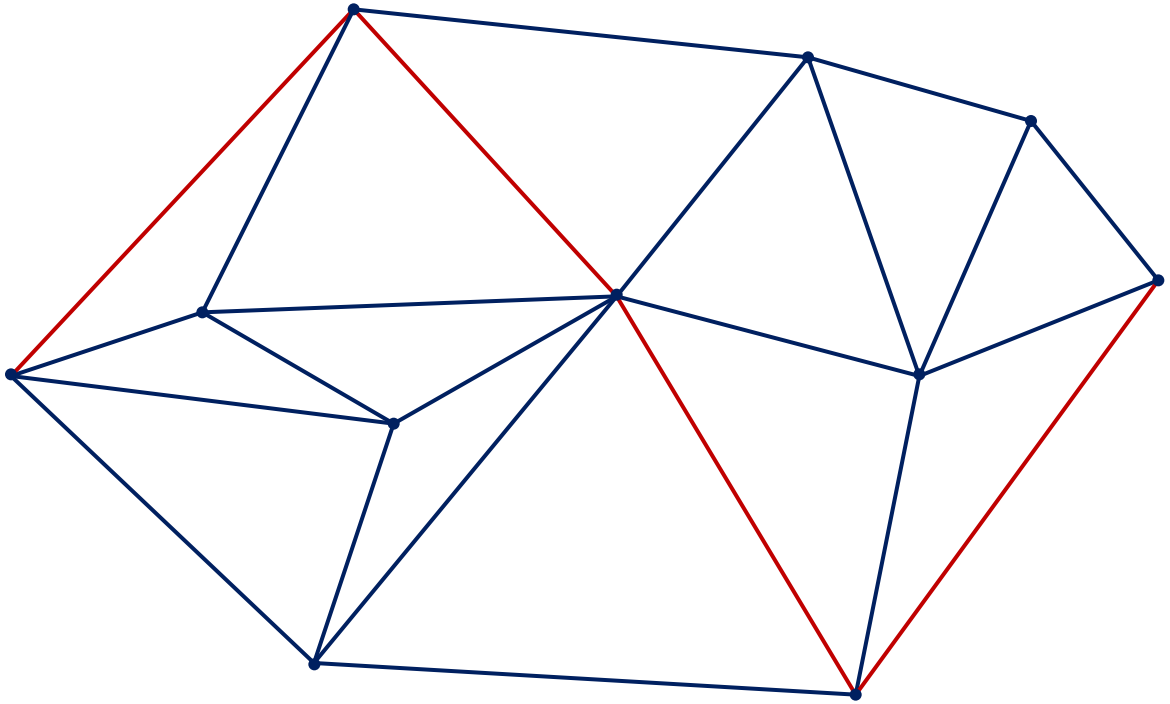


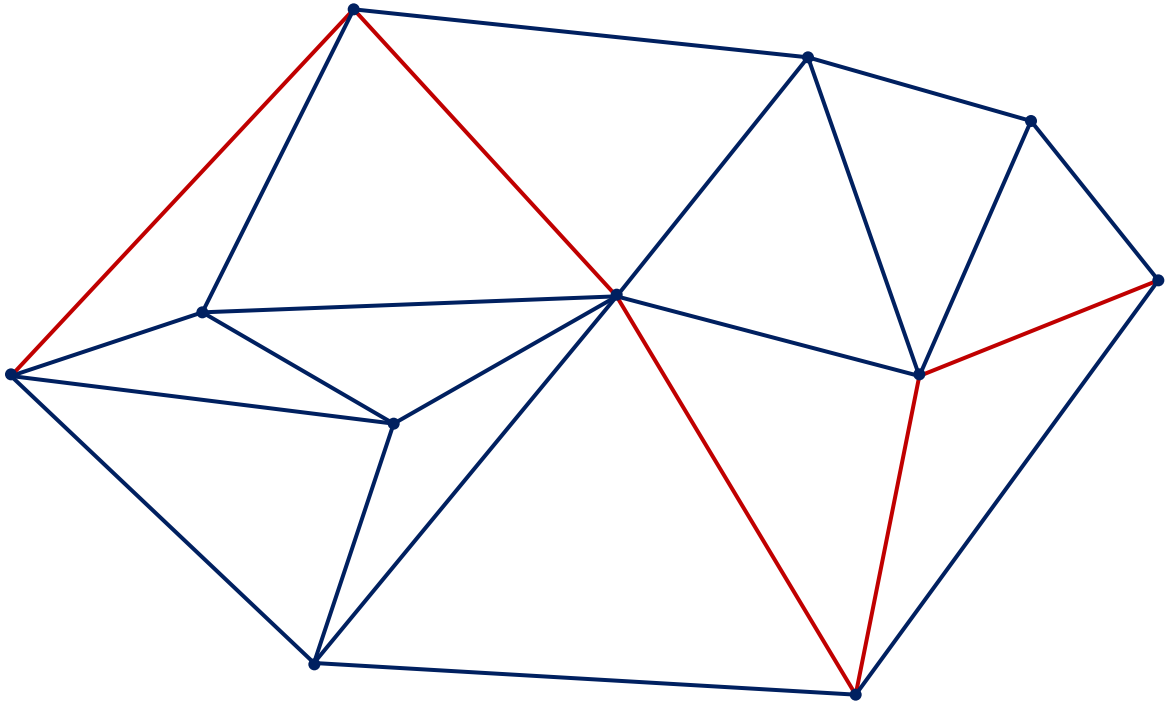


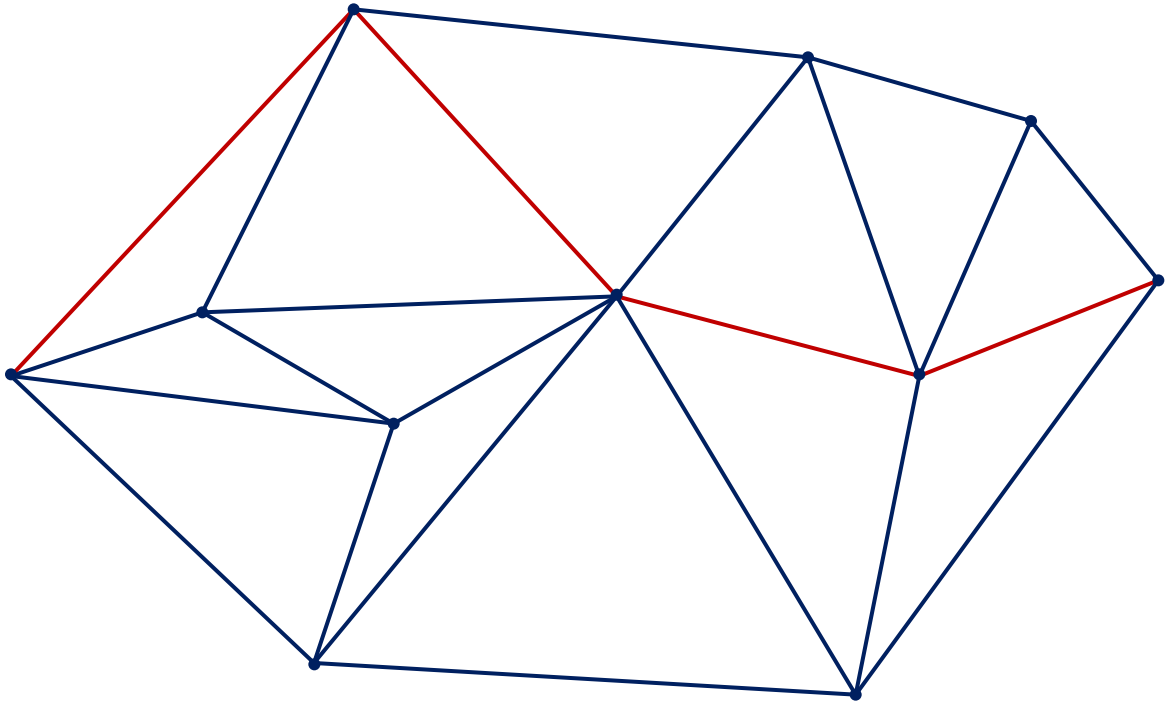


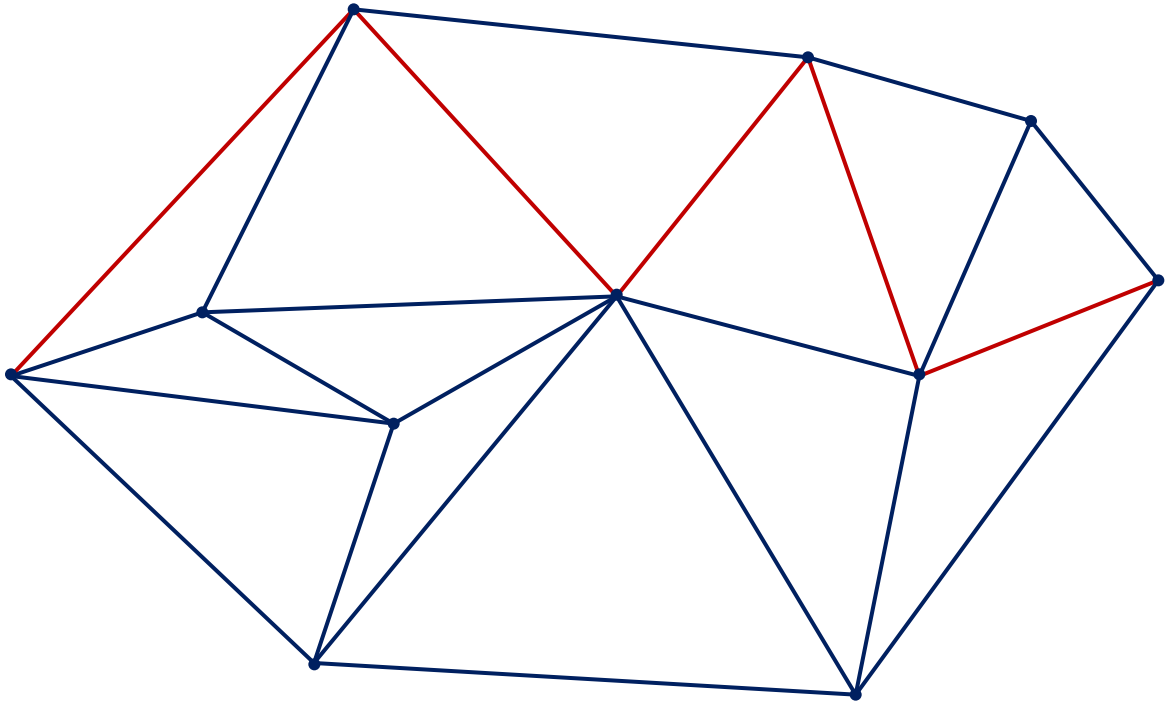


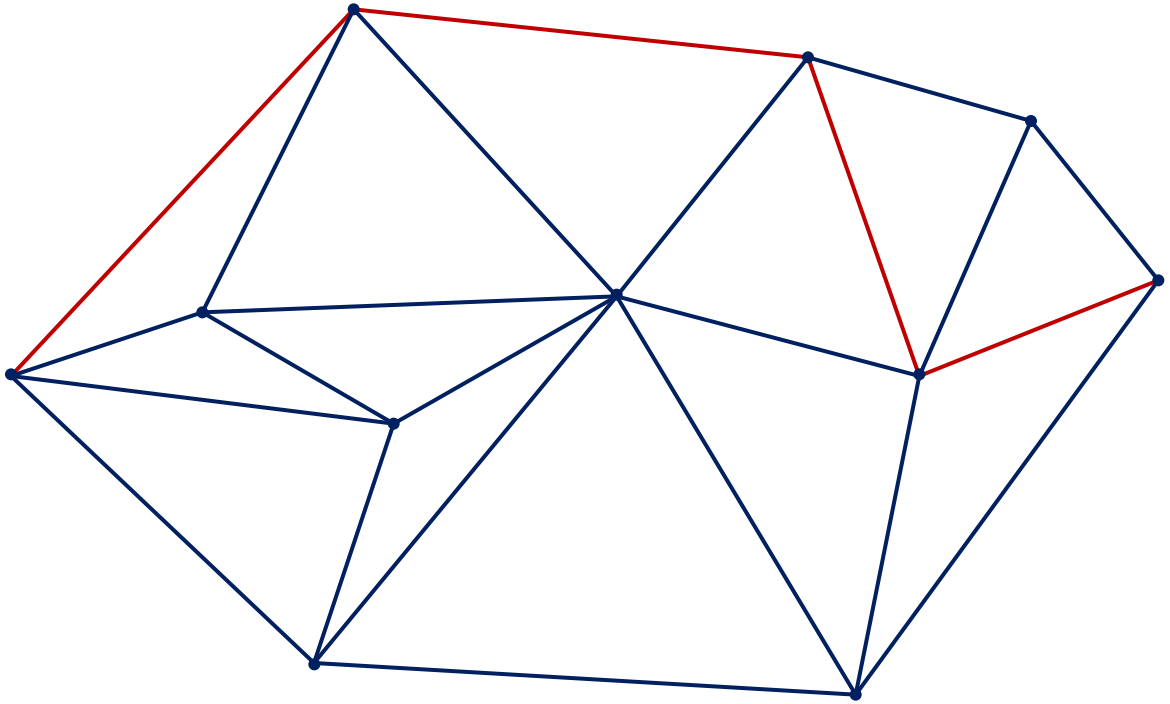


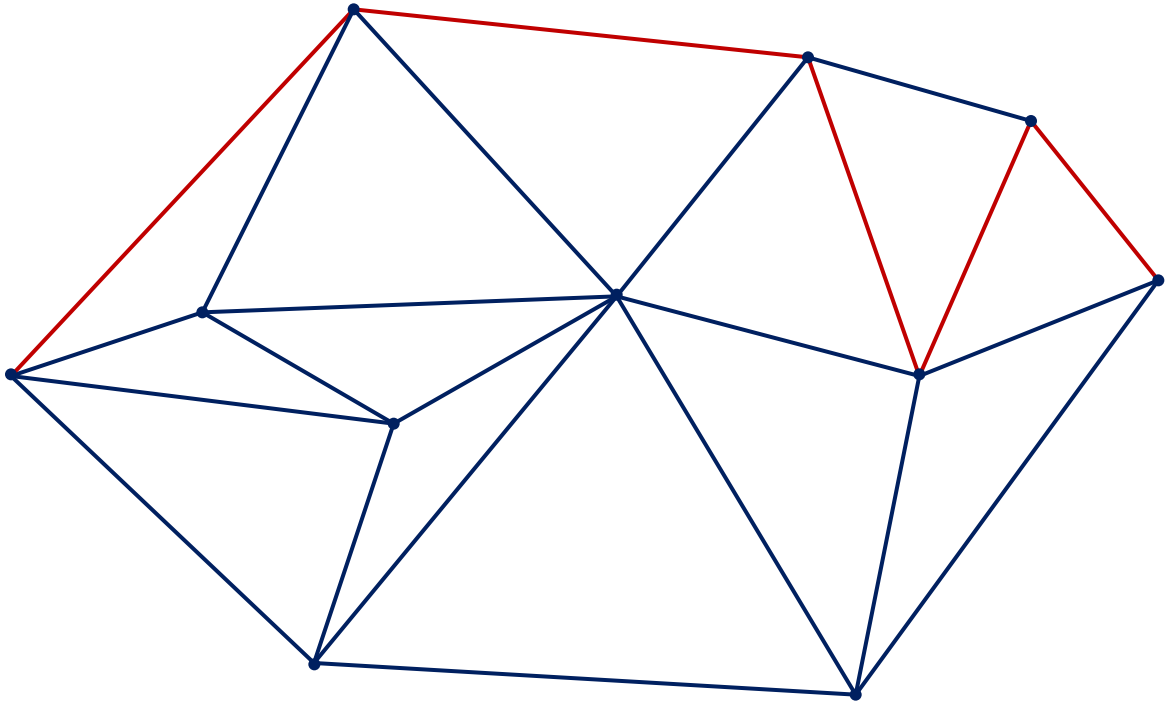


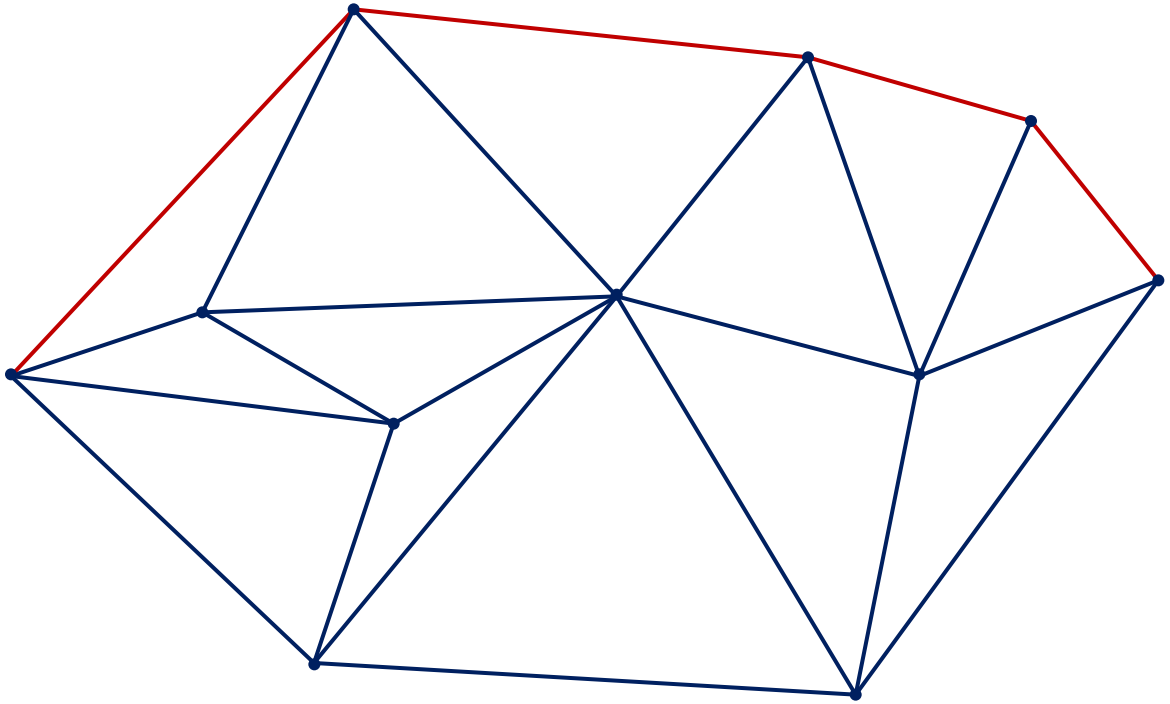












Leftmost advancing monotone chain sweep of a triangulation T

Sequence $C_0, C_1, \ldots, C_t$ of monotone chains of T such that

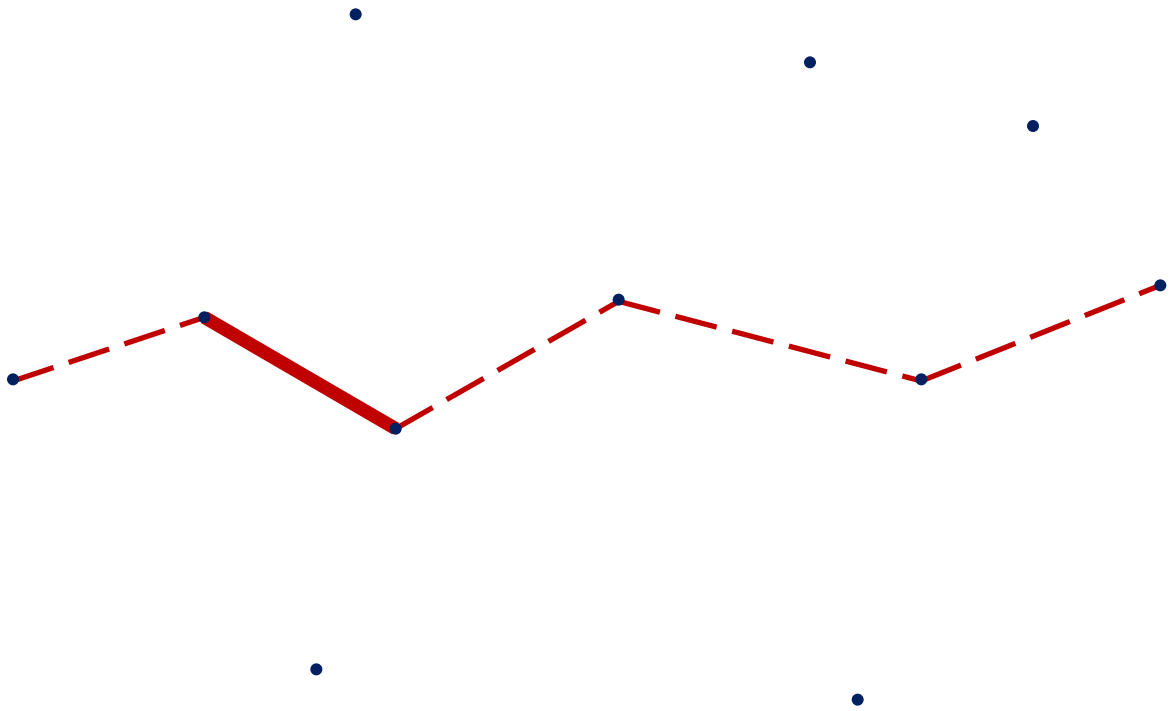
C_0 is the lower convex hull
C_t is the upper convex hull
C_{i+1} obtained from C_i by a leftmost advance

Claim:
Every triangulation has a unique leftmost advancing sweep.


S

S

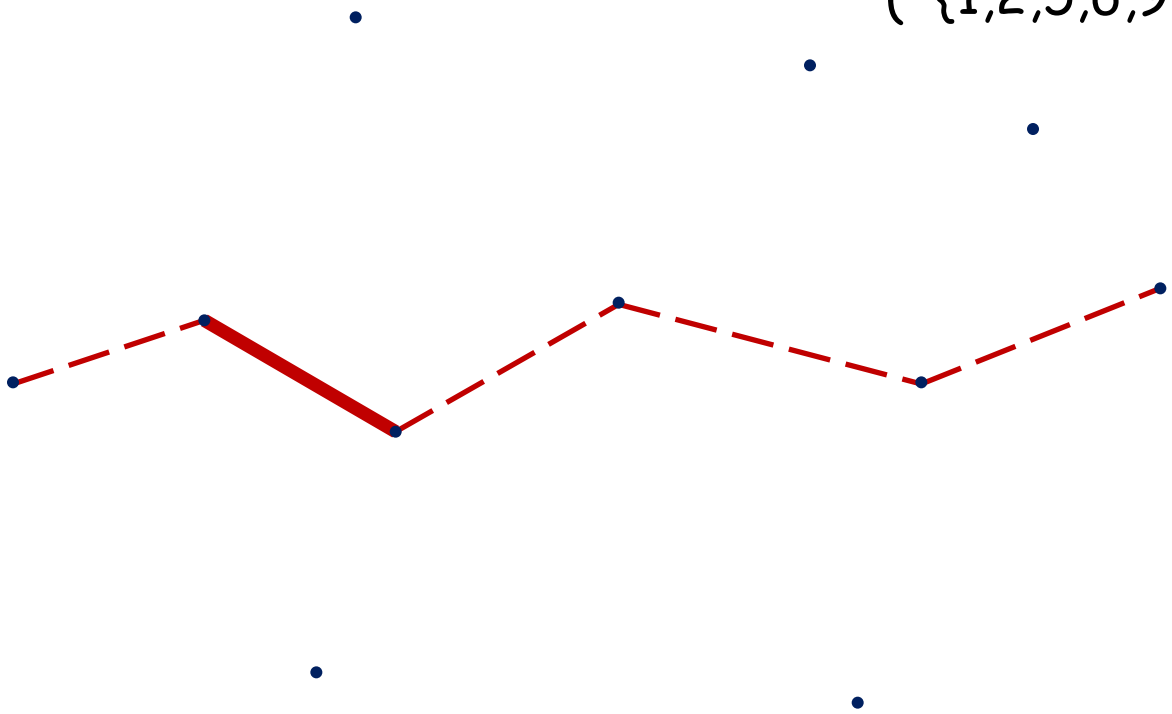
marked monotone chain of S



S

marked monotone chain of S

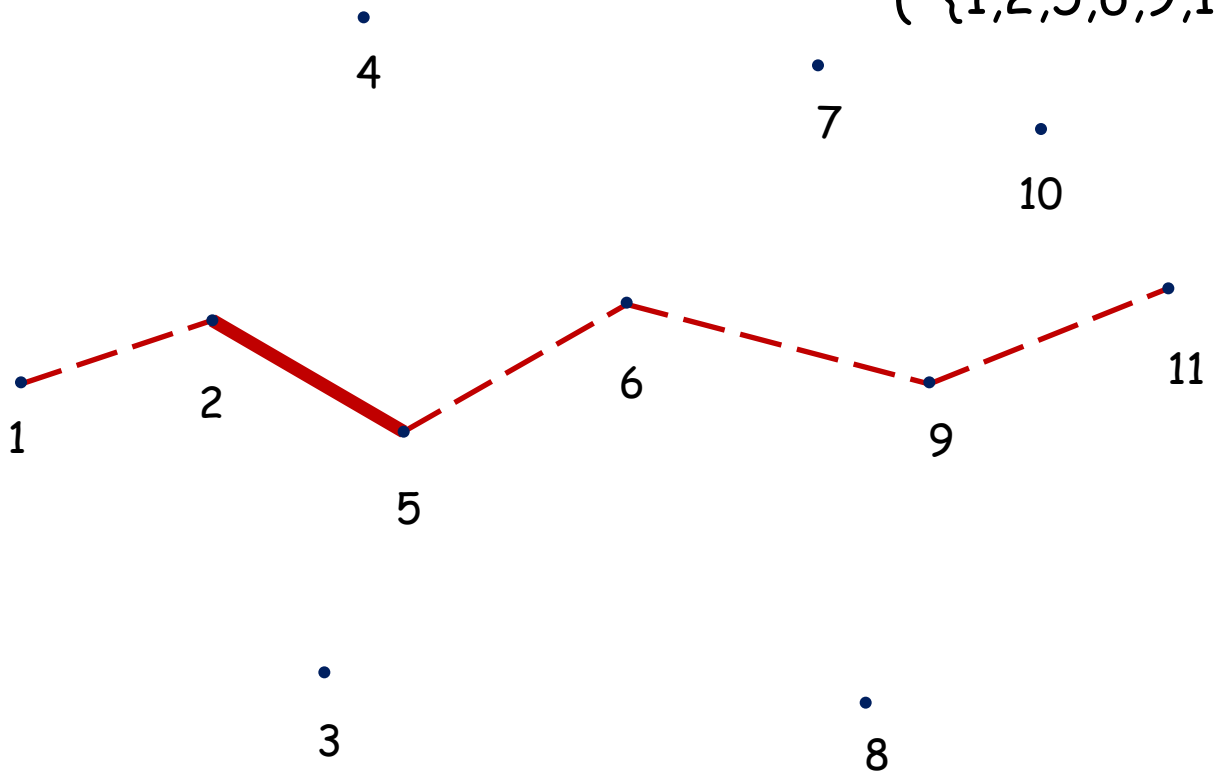
({1,2,5,6,9,11} , 2)



S

marked monotone chain of S

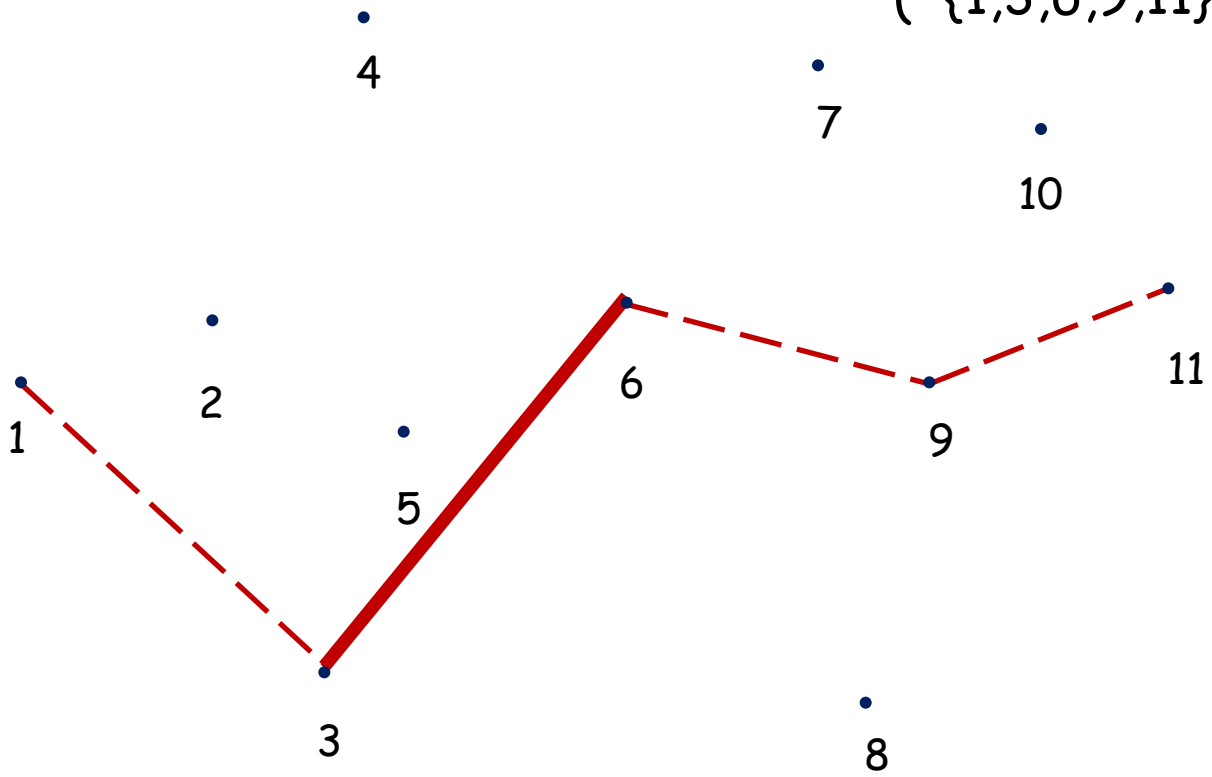
({1,2,5,6,9,11} , 2)



S

marked monotone chain of S

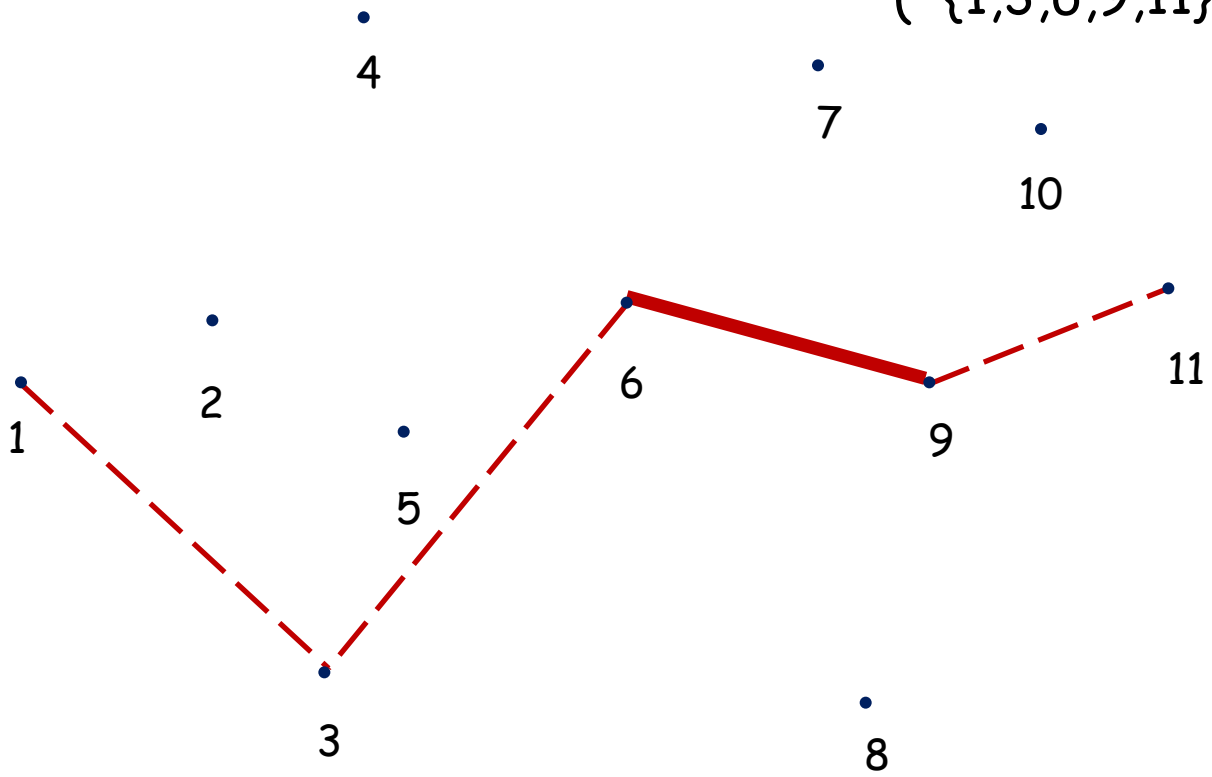
({1,3,6,9,11} , 2)



S

marked monotone chain of S

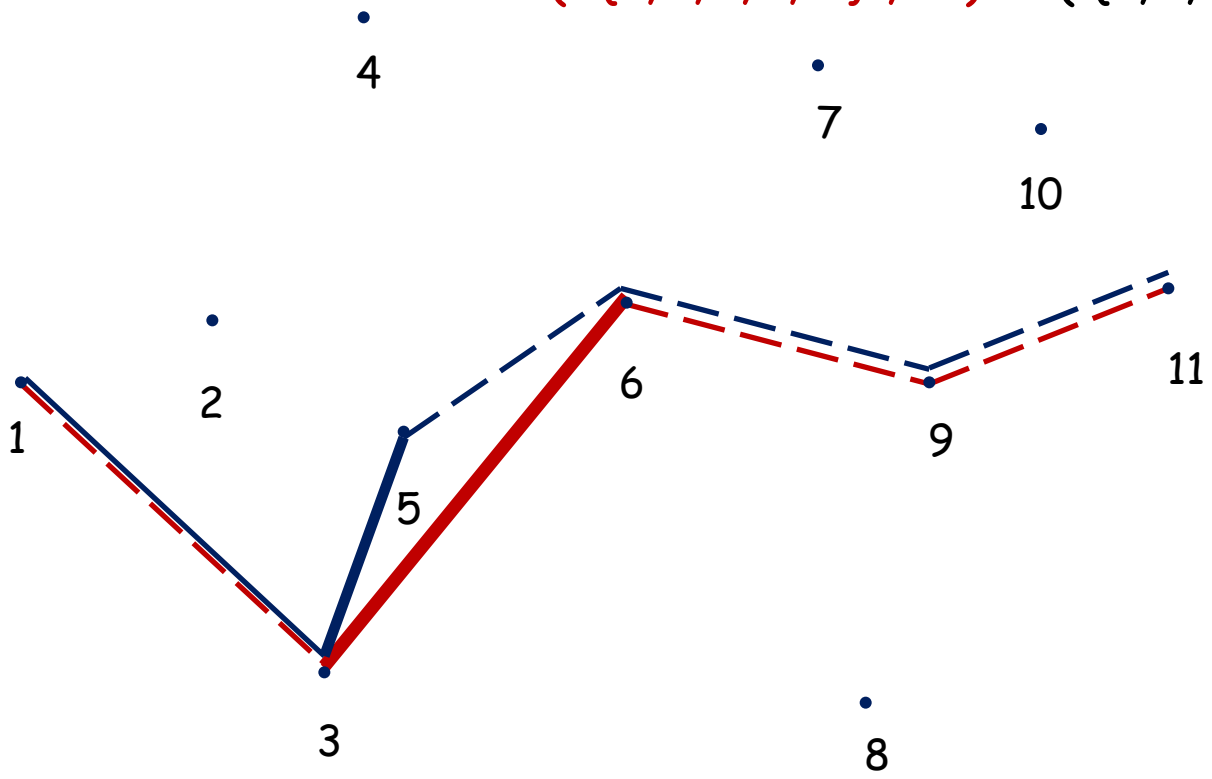
({1,3,6,9,11} , 3)



Successor relation $\prec$ on marked chains

marked monotone chain of S

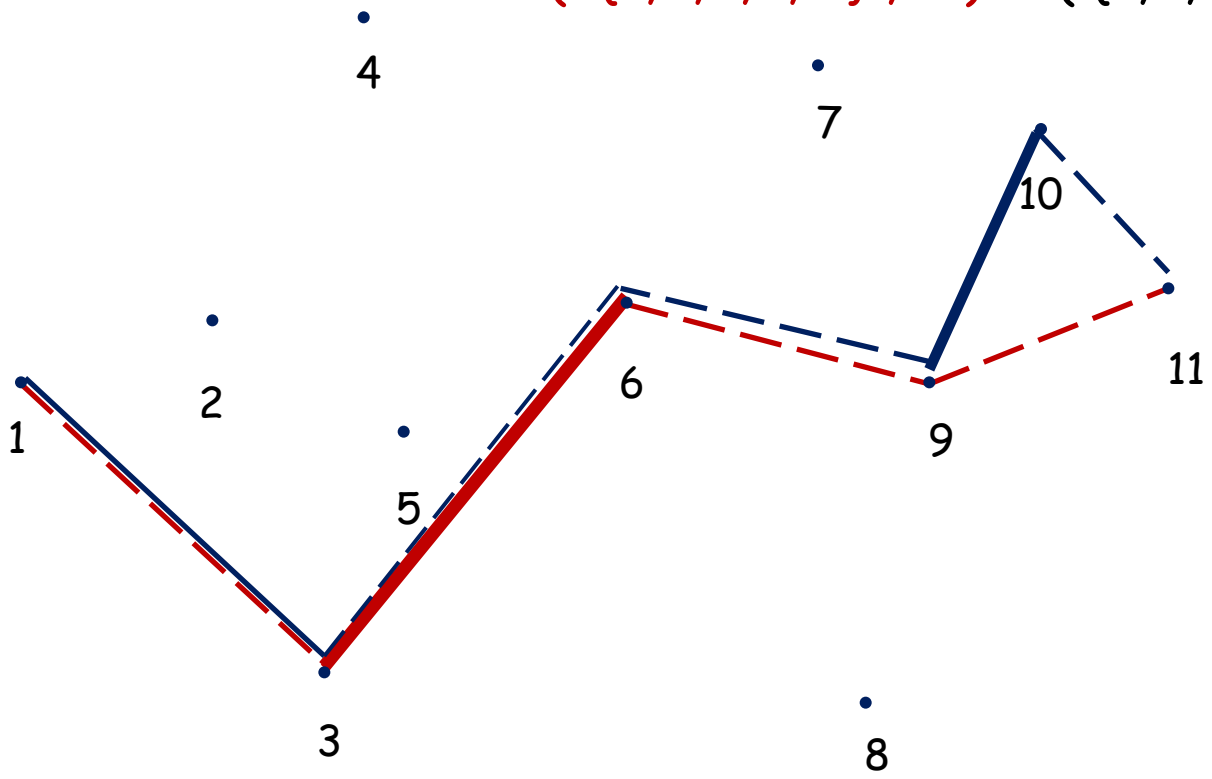
$(\{1,3,6,9,11\} , 2) \prec (\{1,3,5,8,9,11\} , 2)$



Successor relation $\prec$ on marked chains

marked monotone chain of S

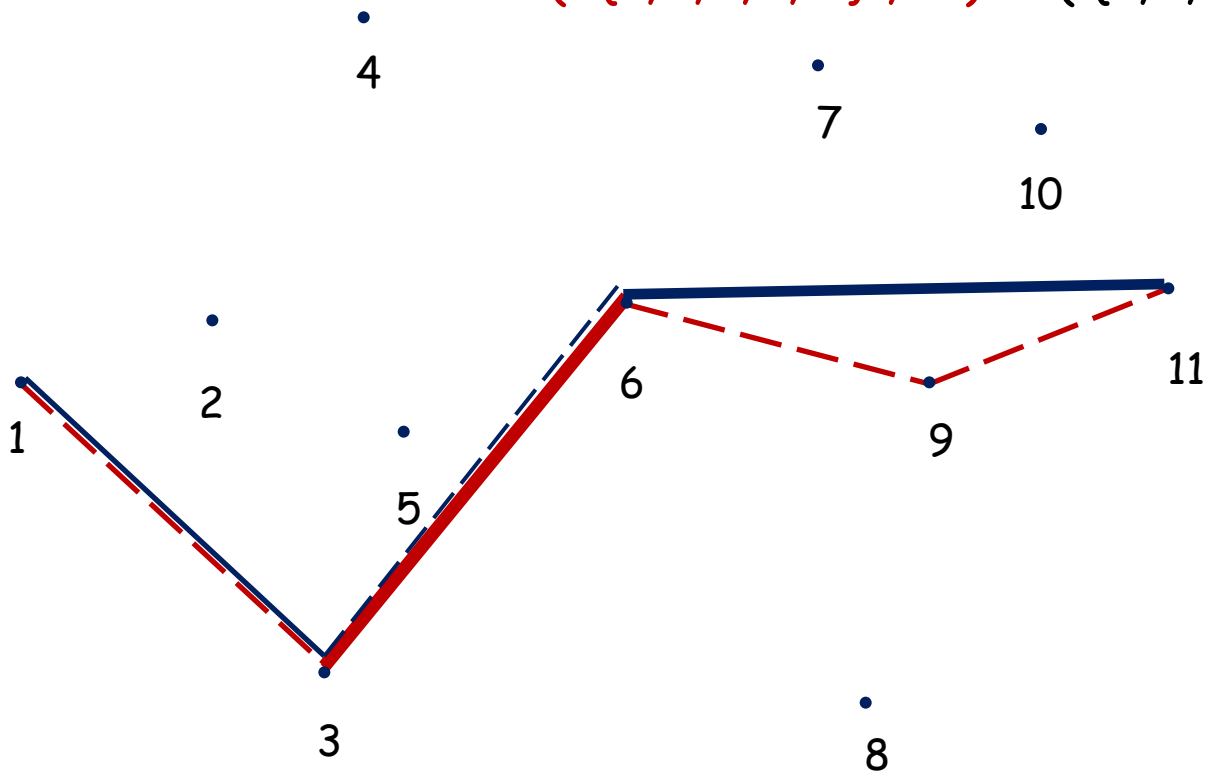
({1,3,6,9,11} , 2) $\prec$ ({1,3,6,9,10,11} , 4)



Successor relation $\prec$ on marked chains

marked monotone chain of S

$(\{1,3,6,9,11\} , 2) \prec (\{1,3,6,11\} , 3)$

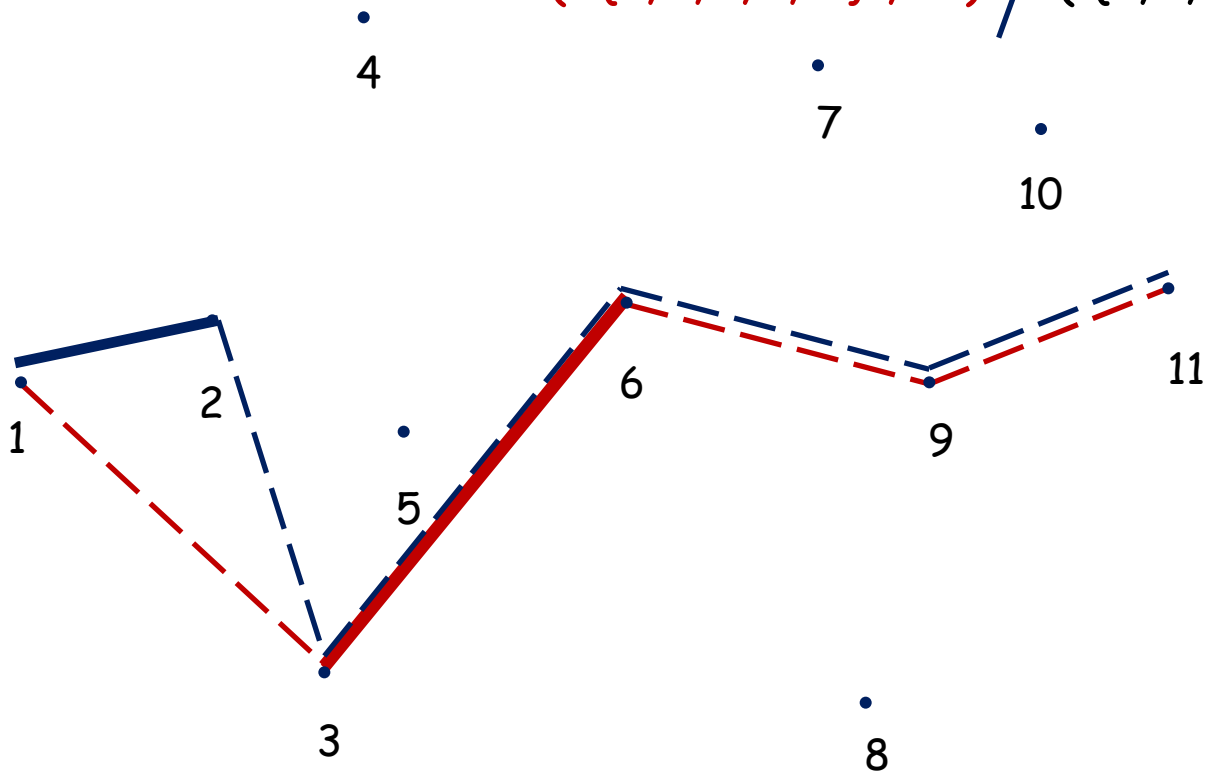


SAARLAND UNIVERSITY
COMPUTER SCIENCE

Successor relation $\prec$ on marked chains

marked monotone chain of S

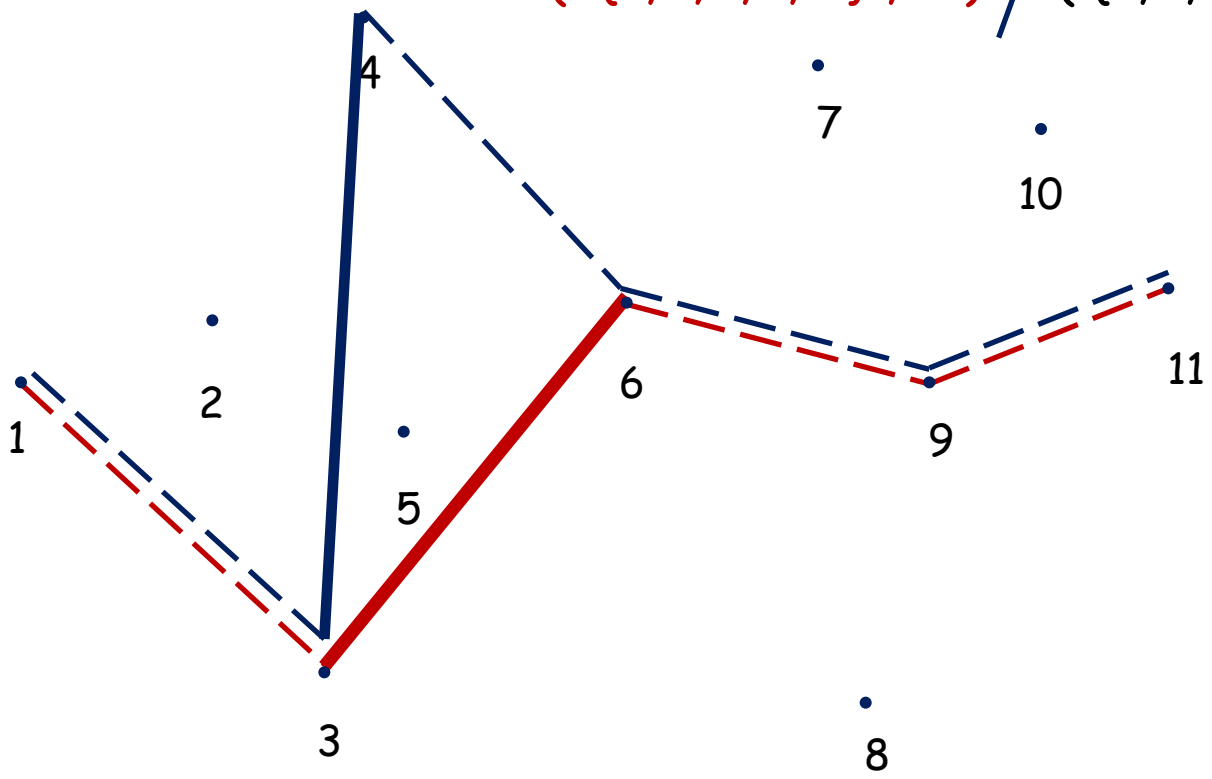
({1,3,6,9,11} , 2) $\not\prec$ ({1,2,3,6,9,11} , 2)



Successor relation $\prec$ on marked chains

marked monotone chain of S

({1,3,6,9,11} , 2) $\not\prec$ ({1,3,4,6,9,11} , 2)



Successor relation $\prec$ on marked chains

Intuition:

(C, k) is to mean that C arose in a leftmost advancing sweep, by an advance that created the k-th edge of C

the next advance must not happen to the left of the k-th edge (otherwise C would not have arisen through a leftmost advance!)

$(C, k) \prec (C', k')$ means that in a leftmost advancing sweep of some triangulation C' could have been created by an advance from C

$G(S)$ Marked monotone chain graph of S

directed acyclic graph
nodes are the marked monotone chains of S
arc from (C, k) to (C', k') iff $(C, k) \prec (C', k')$

$L \subset S$ set of points of S on the lower hull
$U \subset S$ set of points of S on the upper hull

Theorem:
The paths in $G(S)$ from $(L, 1)$ to (U, k), for any k, correspond 1 to 1 with the triangulations of S.

$G(S)$ Marked monotone chain graph of S

directed acyclic graph
nodes are the marked monotone chains of S
arc form (C, k) to (C', k') iff $(C, k) \prec (C', k')$

$L \subset S$ set of points of S on the lower hull
$U \subset S$ set of points of S on the upper hull

Theorem:
The paths in $G(S)$ from $(L, 1)$ to TOP, correspond 1 to 1 with the triangulations of S,
where TOP is an additional vertex of $G(S)$ with arcs leading into it from (U, k) for every k.

Theorem:
The paths in $G(S)$ from $(L, 1)$ to TOP, correspond 1 to 1 with the triangulations of S.

Corollary:
The number of triangulations of S can be determined by computing the number of paths in $G(S)$ from $(L, 1)$ to TOP.

Since $G(S)$ is a DAG this can be done
in time linear in the size of $G(S)$, which is $O(2^n n^2)$
and in space linear in the number of vertices of $G(S)$,
which is $O(n2^n)$.

Implementation

Implementation

can handle random point sets up to size 50, which was out of reach before

Implementation

can handle random point sets up to size 50, which was out of reach before

comparatively slow on some point sets, e.g. convex position

Implementation

can handle random point sets up to size 50, which was out of reach before

comparatively slow on some point sets, e.g. convex position

Very space intensive!!

- 128 Gbyte main memory machine
- do not store entire $G(S)$
- run algorithm several times on the same input,
  - count modulo different primes, and use Chinese Remaindering

Easy Generalizations

Count triangulations of S that are constrained to contain certain edges.

Easy Generalizations

Count triangulations of S that are constrained to contain certain edges.

Truly uniform random generation of triangulations of a given set S.

Easy Generalizations

Count triangulations of S that are constrained to contain certain edges.

Truly uniform random generation of triangulations of a given set S.

Fast unbiased estimation of tr(S).

Truly uniform random generation of a triangulation of S

1. Preprocessing:
   Generate $G(S)$
   with each node v store
   $p(v)$ = #paths from source to v
   with each node v store the prefix sums
   of the sequence of $p(u_1),\ldots,p(u_k)$,
   where $u_1,\ldots,u_k$ are the predecessor of v in $G(S)$

   store $Z = p(TOP) = tr(S)$

2. Sampling:
   Generate a random number Y in $\{1,\ldots,Z\}$
   Using the info computed in 1. determine the Y'th source-sink path in $G(S)$ (in reverse order)

Truly uniform random generation of a triangulation of S

Theorem:
After preprocessing that takes time $O(2^n n^2)$ and uses space $O(2^n n^2)$,
triangulations of S can be generated truly uniformly at random in
time $O(n \log n)$ per sample.

Truly uniform random generation of a triangulation of S

Theorem:
After preprocessing that takes time $O(2^n n^2)$ and uses space $O(2^n n^2)$,
triangulations of S can be generated truly uniformly at random in
time $O(n \log n)$ per sample.

Why is this useful ???

Estimating the proportion of triangulations of S that are **regular**

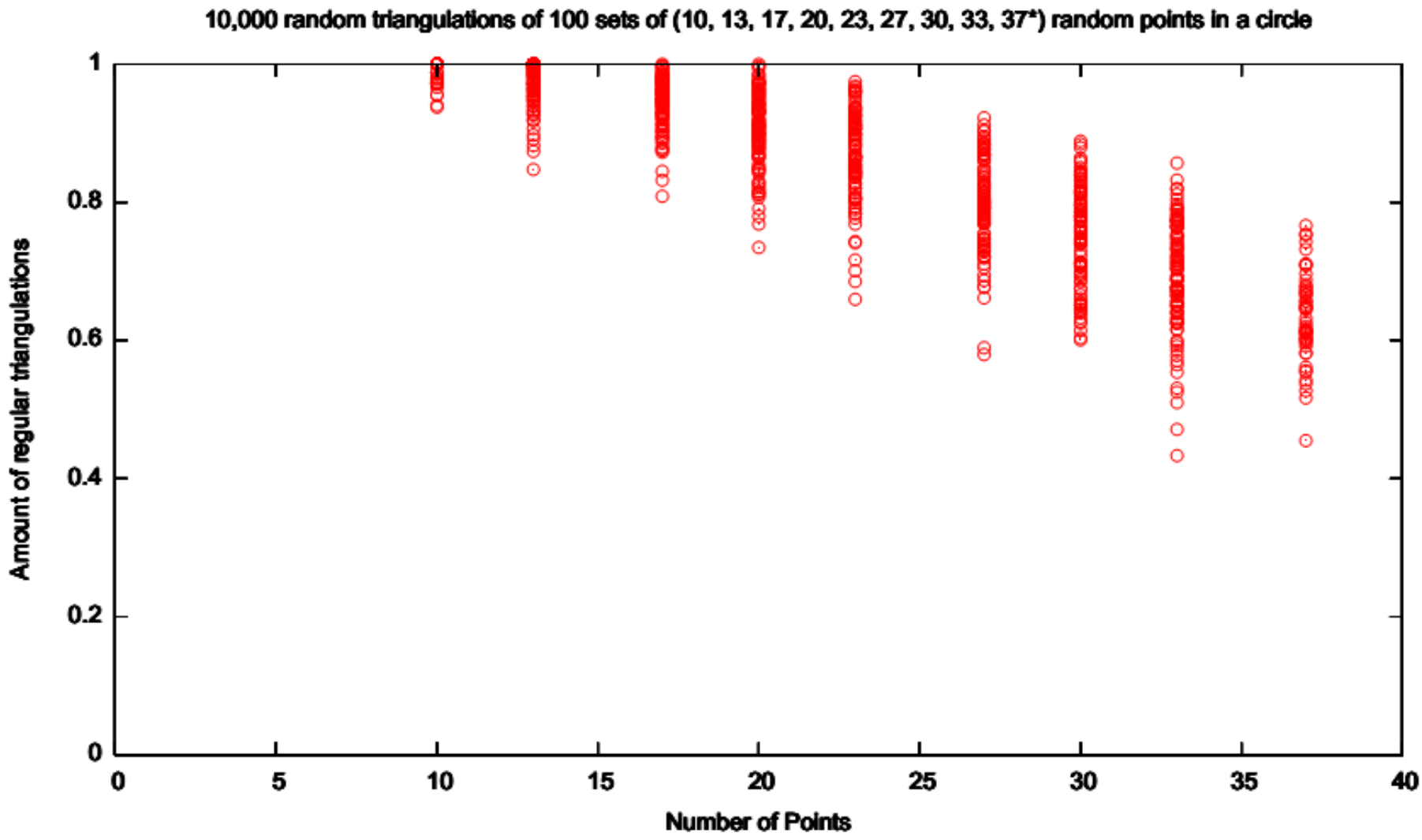
Estimating the proportion of triangulations of S that are **regular**

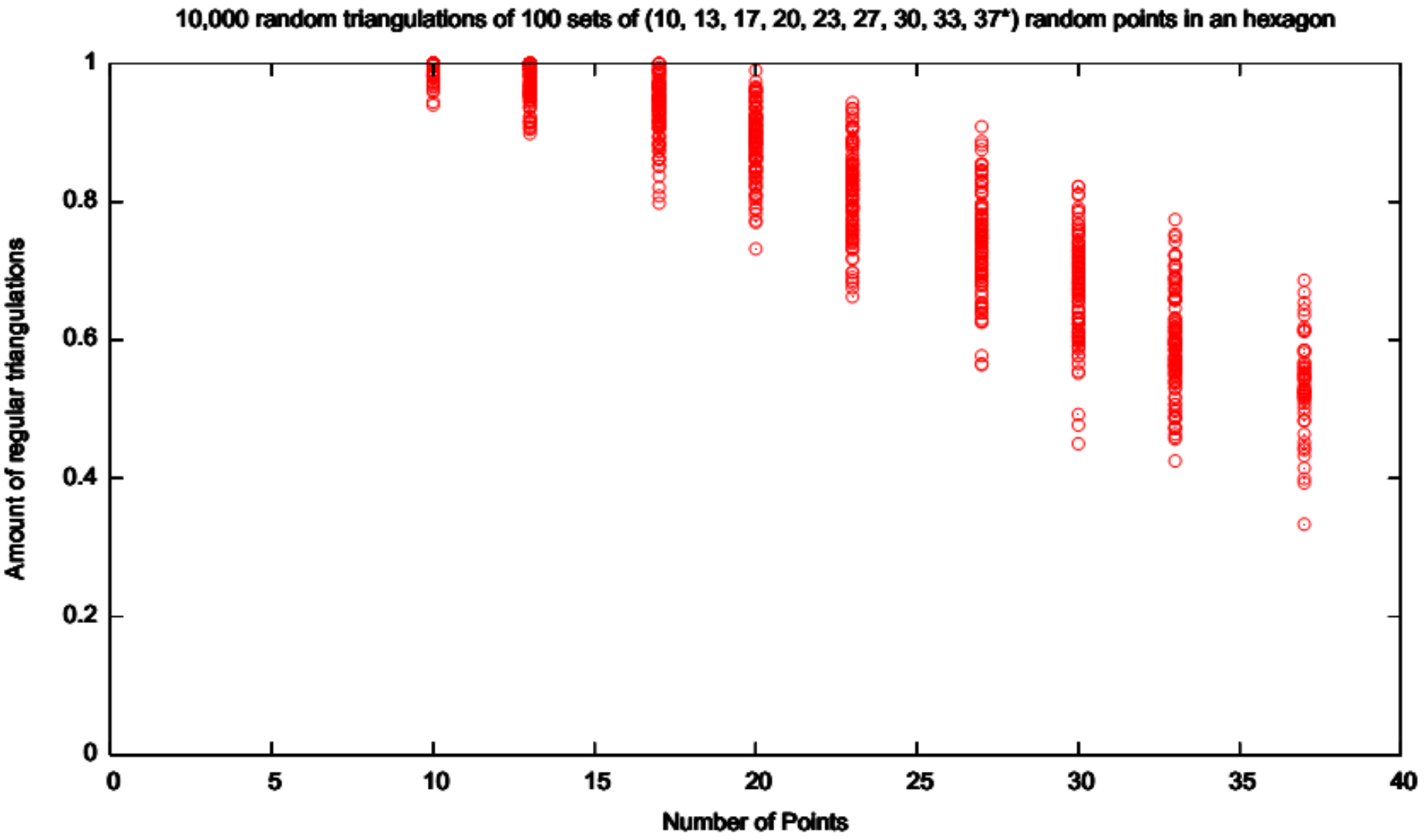
triangulation T of S is regular means

> S can be lifted to a set S′ in $\mathbb{R}^3$ so that T is the projection of the lower convex hull of S′

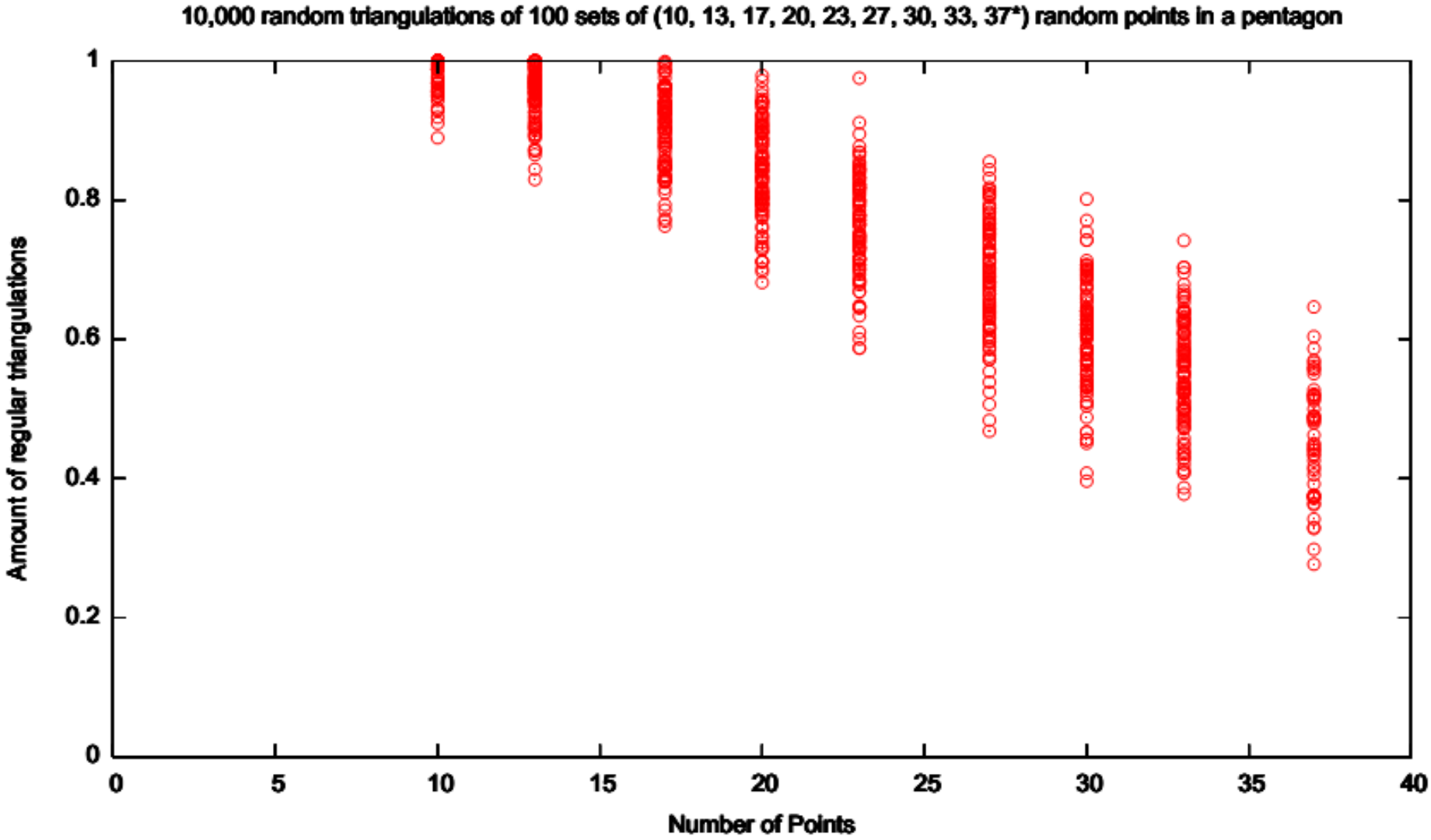
Regularity of a triangulation can be tested by checking a certain derived LP for feasibility.

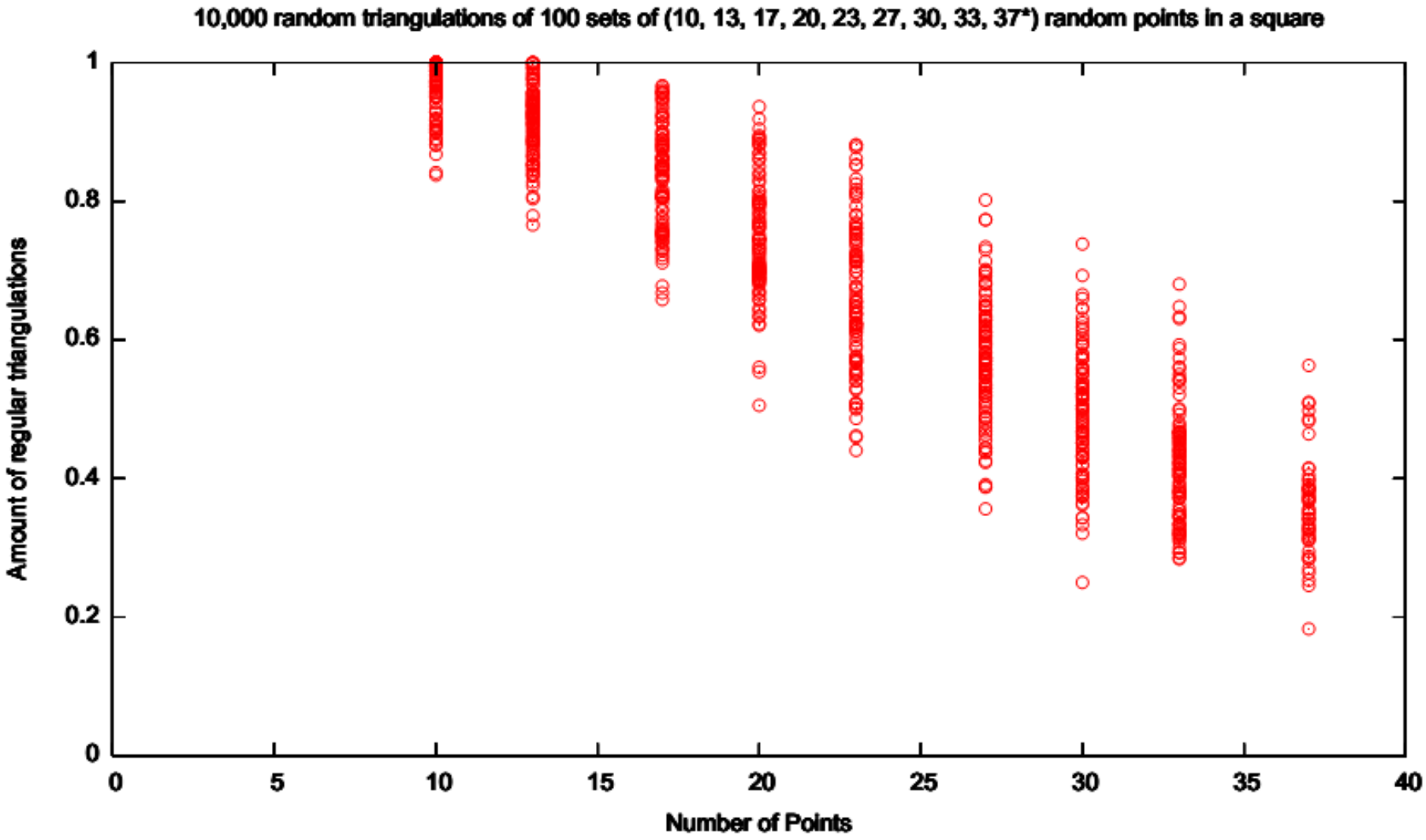
Estimate the proportion of the triangulations of S that are regular by repeated random sampling and testing.

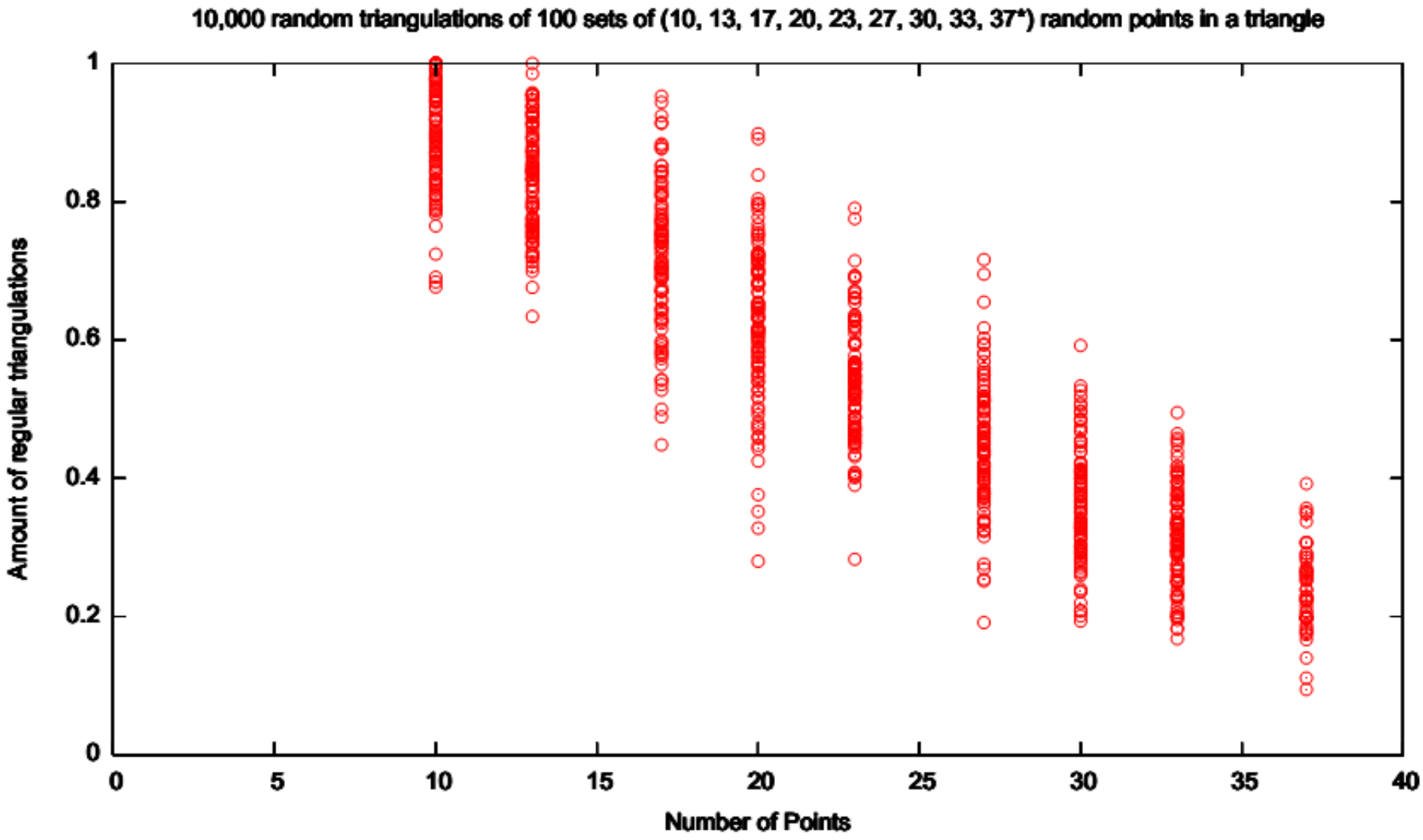


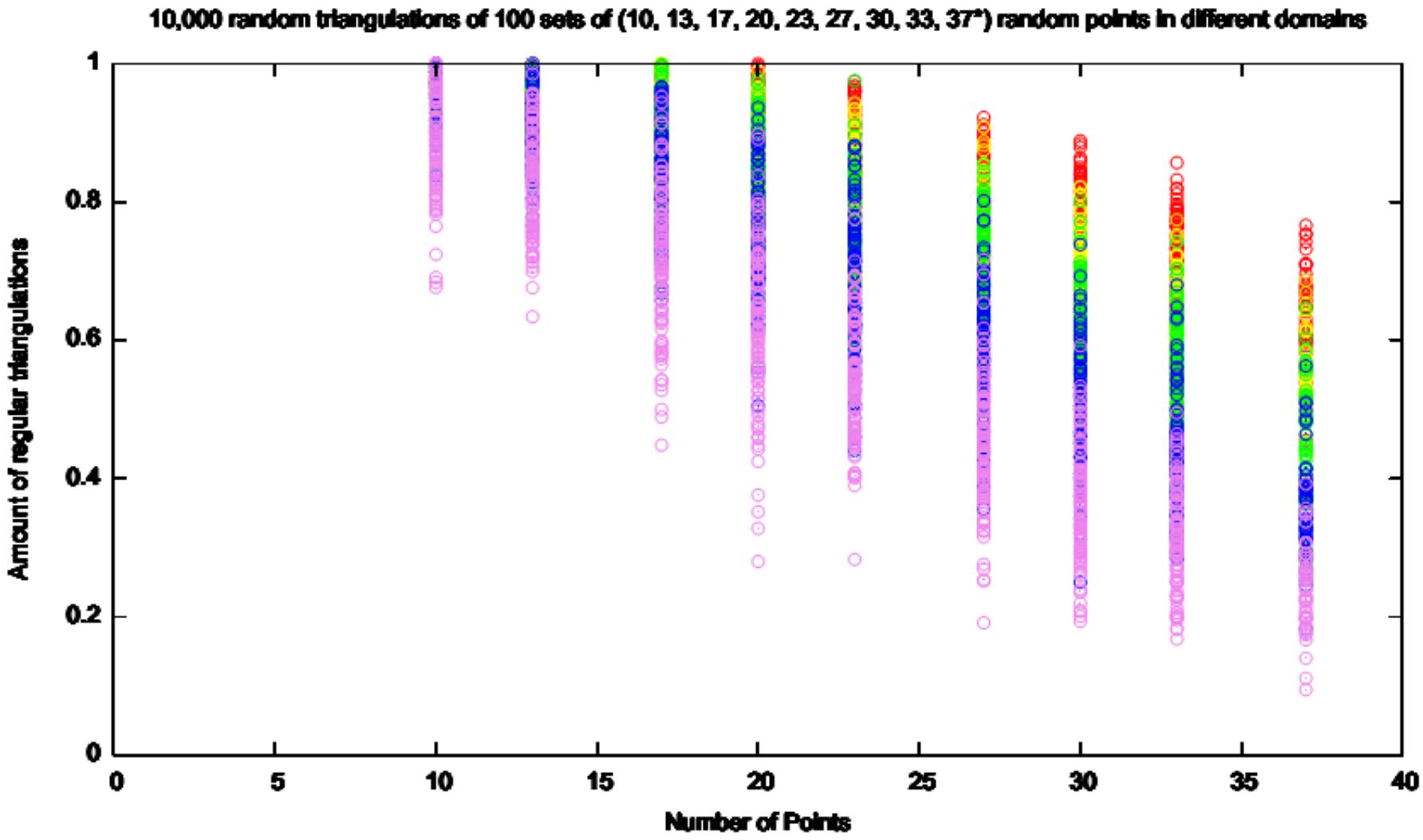


SAARLAND UNIVERSITY
COMPUTER SCIENCE









SAARLAND UNIVERSITY
COMPUTER SCIENCE

Easy Generalizations

Count triangulations of S that are constrained to contain certain edges.

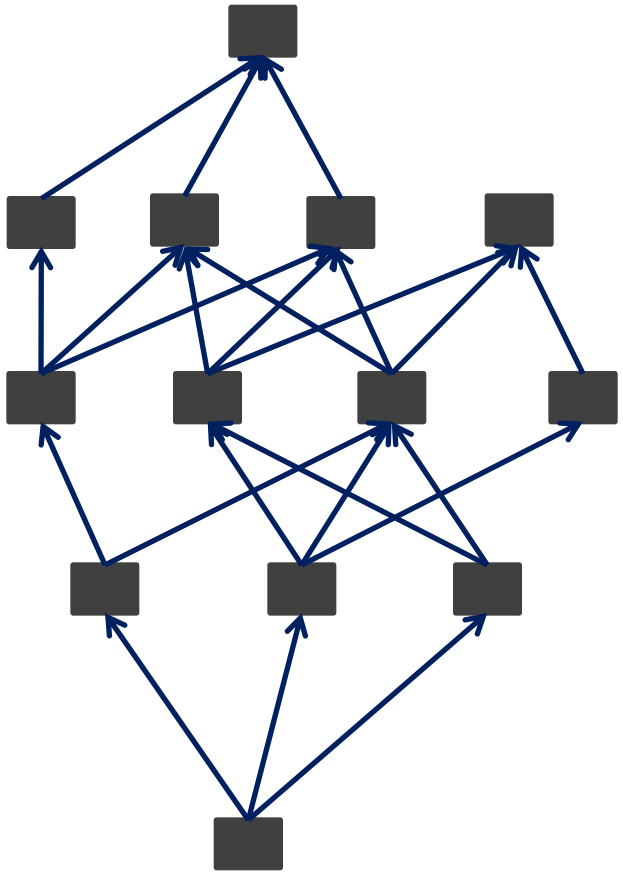
Truly uniform random generation of triangulations of a given set S.

Fast unbiased estimation of tr(S).

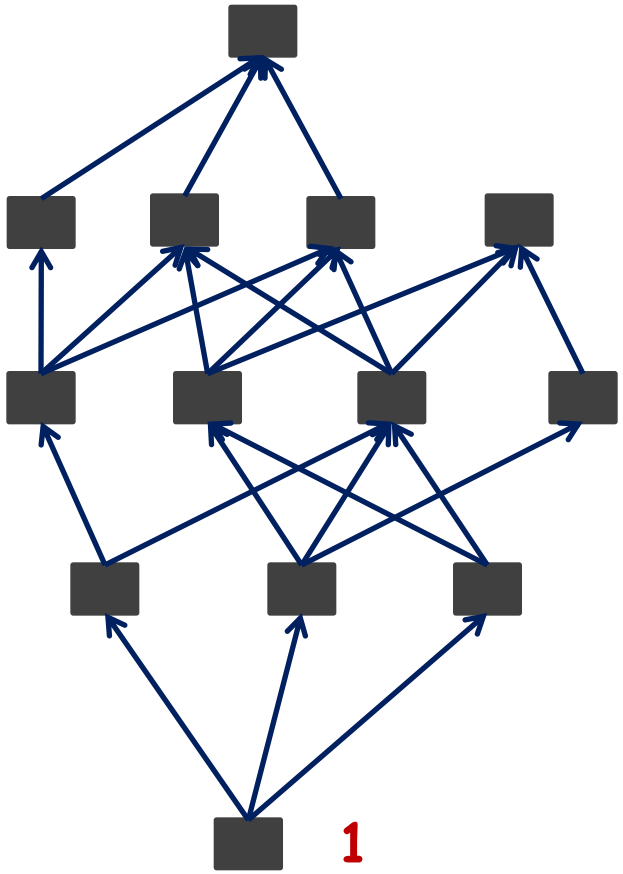
Fast unbiased estimation of tr(S)

•

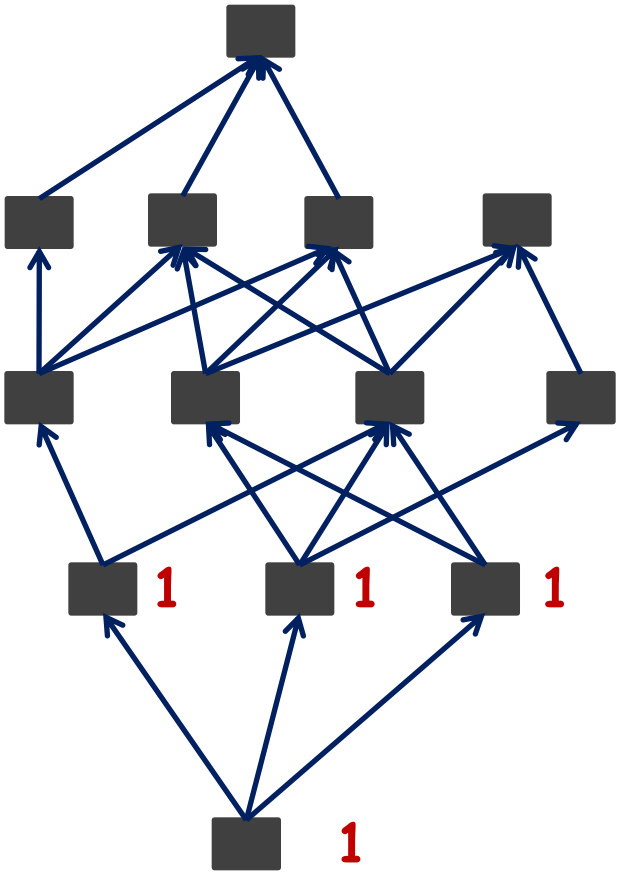
Estimating the number of source-sink paths in a graded DAG



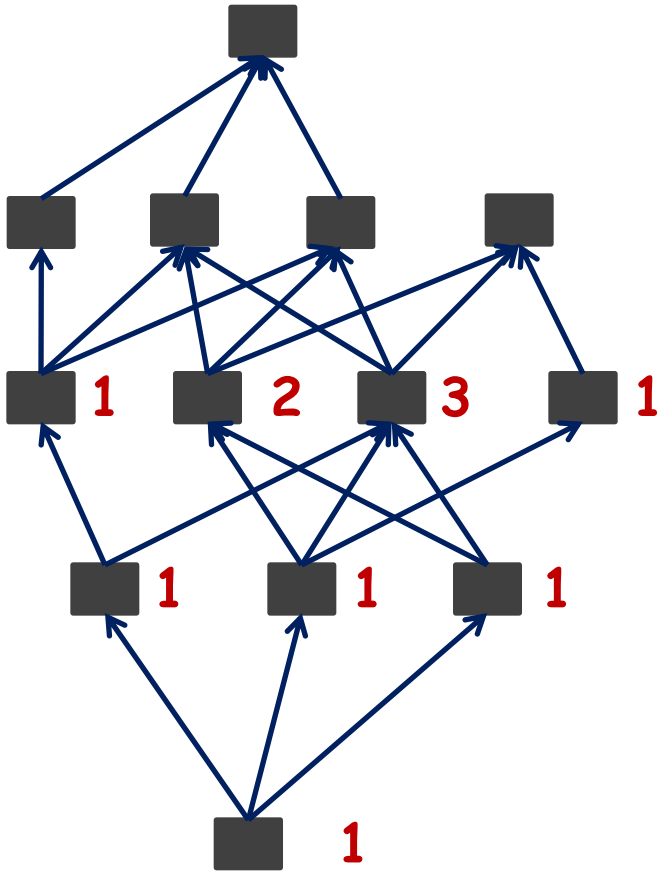
Estimating the number of source-sink paths in a graded DAG



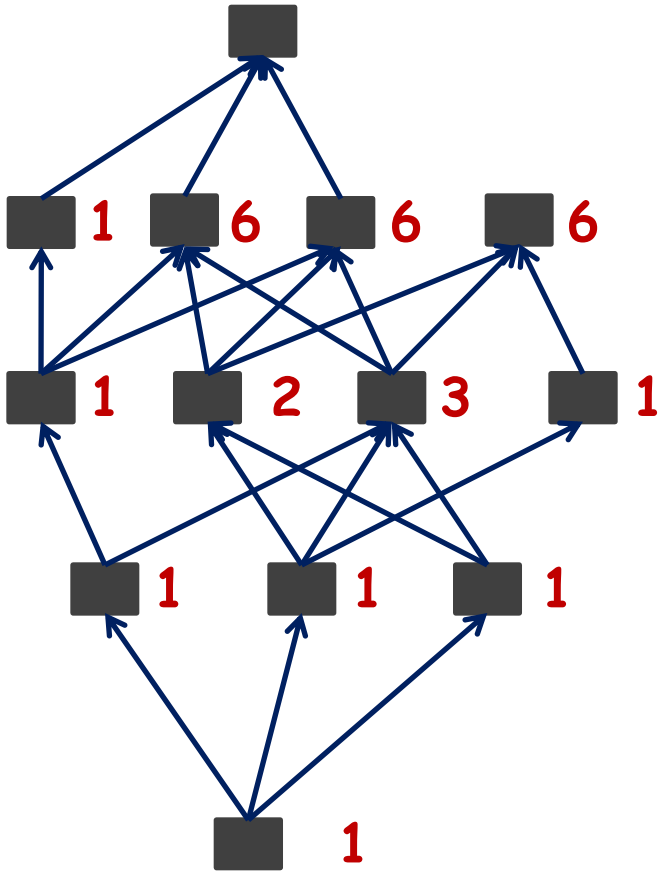
Estimating the number of source-sink paths in a graded DAG



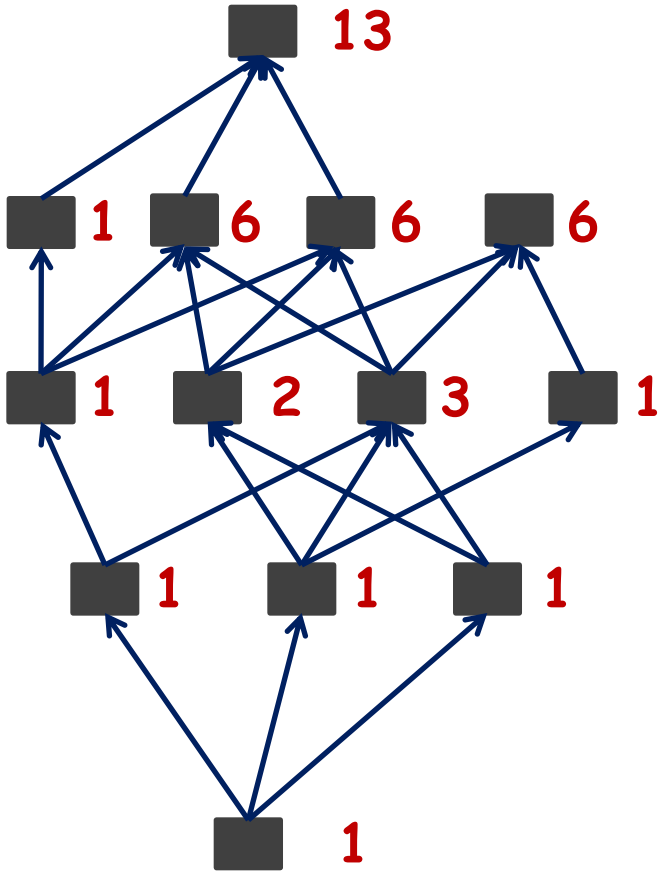
Estimating the number of source-sink paths in a graded DAG



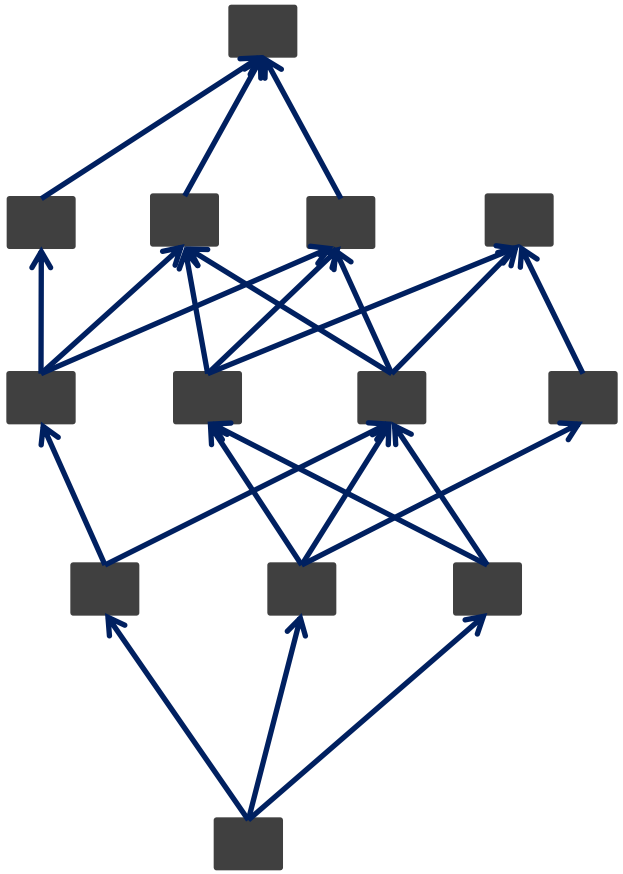
Estimating the number of source-sink paths in a graded DAG



Estimating the number of source-sink paths in a graded DAG

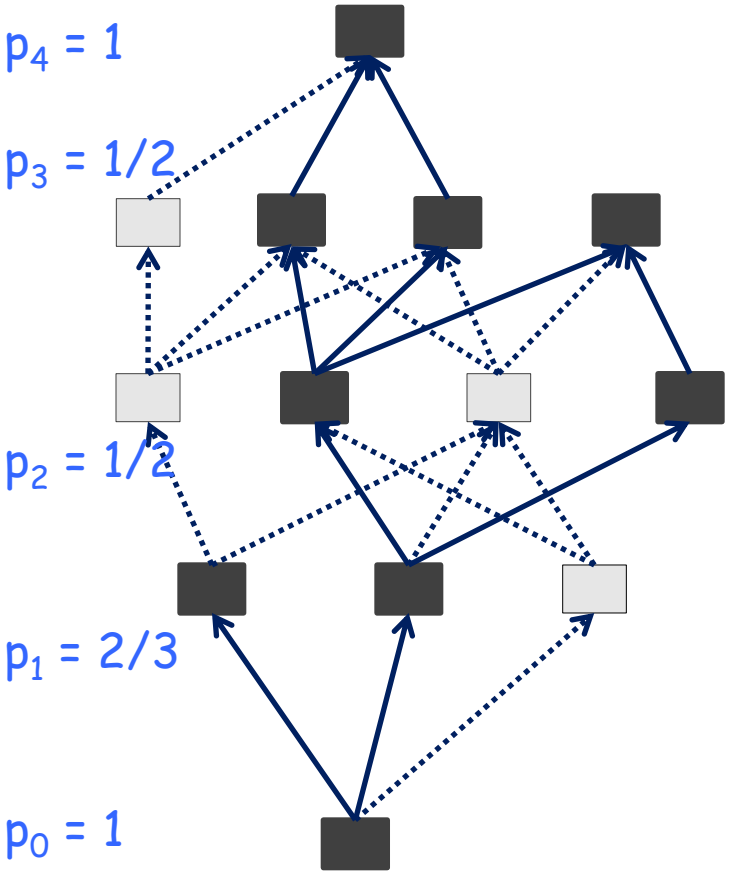


Estimating the number of source-sink paths in a graded DAG



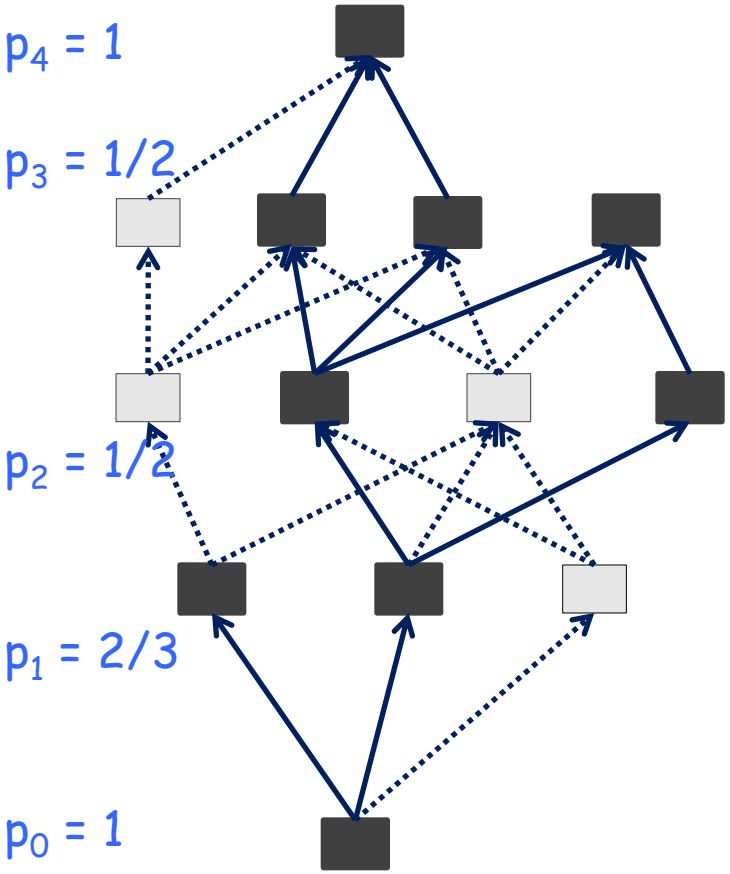
on level i keep each node independently with probability p_i

Estimating the number of source-sink paths in a graded DAG



on level i keep each node independently with probability p_i

Estimating the number of source-sink paths in a graded DAG



on level i keep each node independently with probability p_i

X = # surviving souce-sink paths

X_w = 1 iff path w survives, 0 ow.

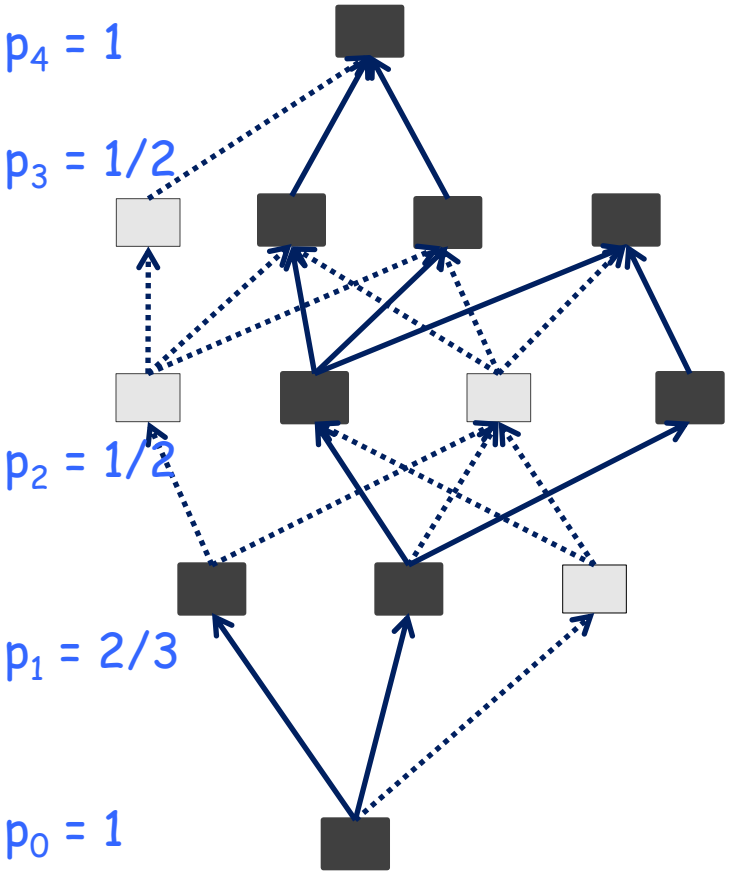
$X = \sum_w X_w$ $\qquad$ $Ex[X] = \sum_w Ex[X_w]$

$Ex[X_w]$ = Pr(path w survives) = $\prod_i p_i = P$

$Ex[X] = \sum_w P = Z\,P$

where Z total # of paths

Estimating the number of source-sink paths in a graded DAG



$Z = Ex[X] / P$

on level i keep each node independently with probability p_i

X = # surviving souce-sink paths

X_w = 1 iff path w survives, 0 ow.

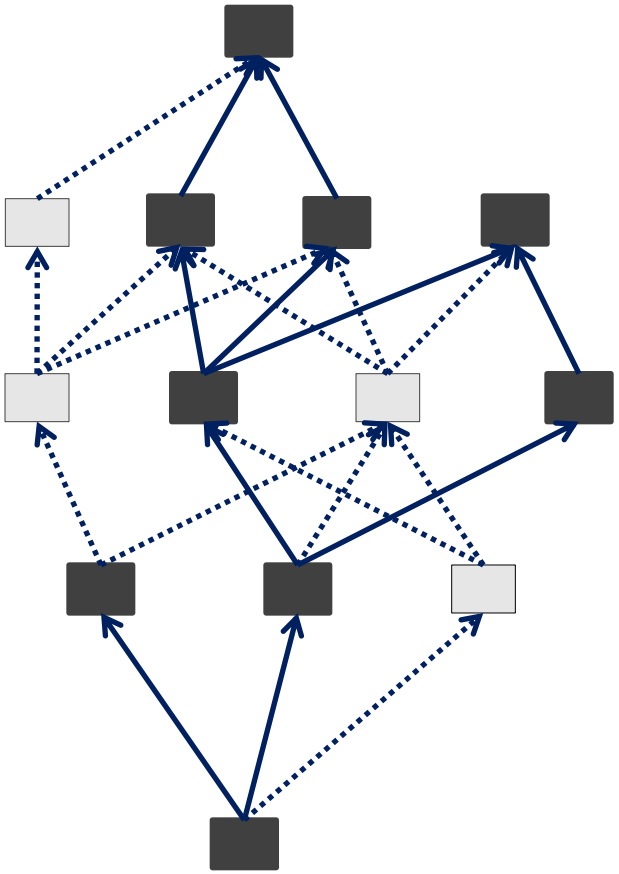
$X = \sum_w X_w$ $\qquad$ $Ex[X] = \sum_w Ex[X_w]$

$Ex[X_w]$ = Pr(path w survives) = $\prod_i p_i = P$

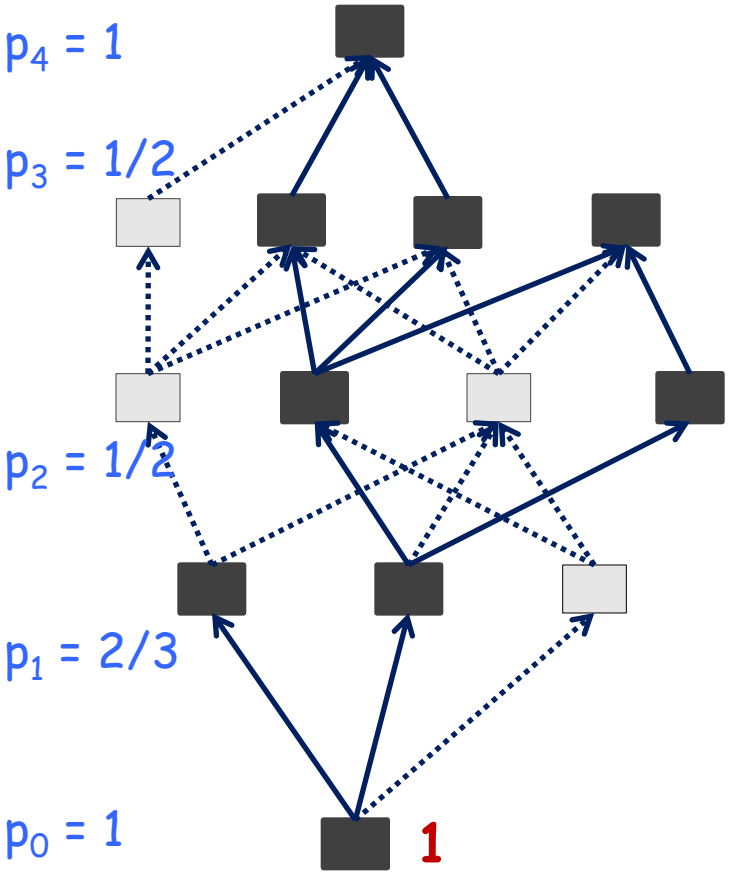
$Ex[X] = \sum_w P = Z P$

where Z total # of paths

Estimating the number of source-sink paths in a graded DAG



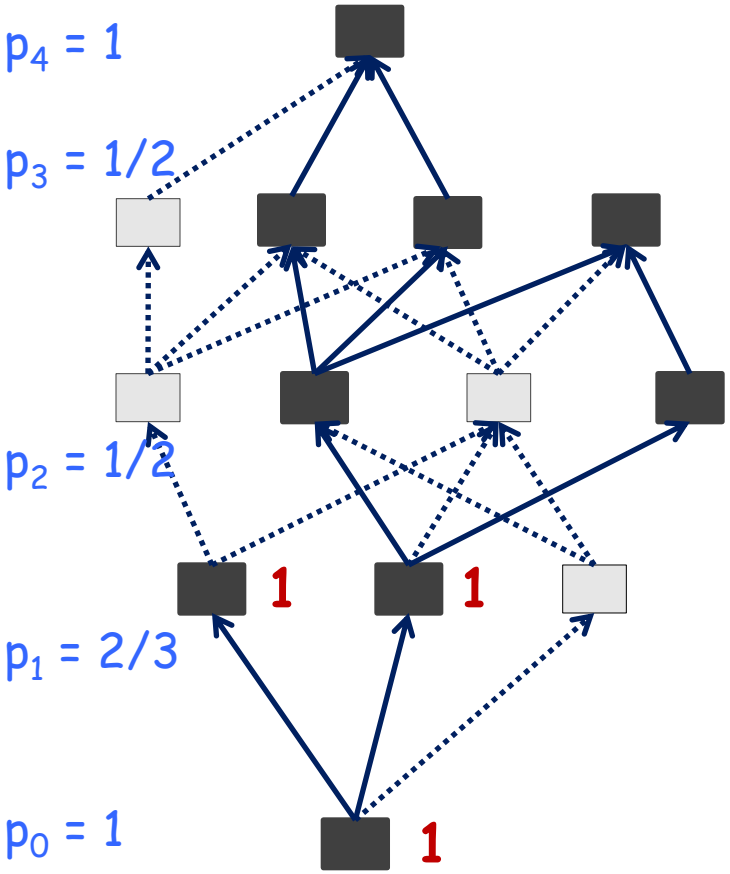
Estimating the number of source-sink paths in a graded DAG



$$Z = Ex[X] / P$$

$$P = \Pi_i p_i$$

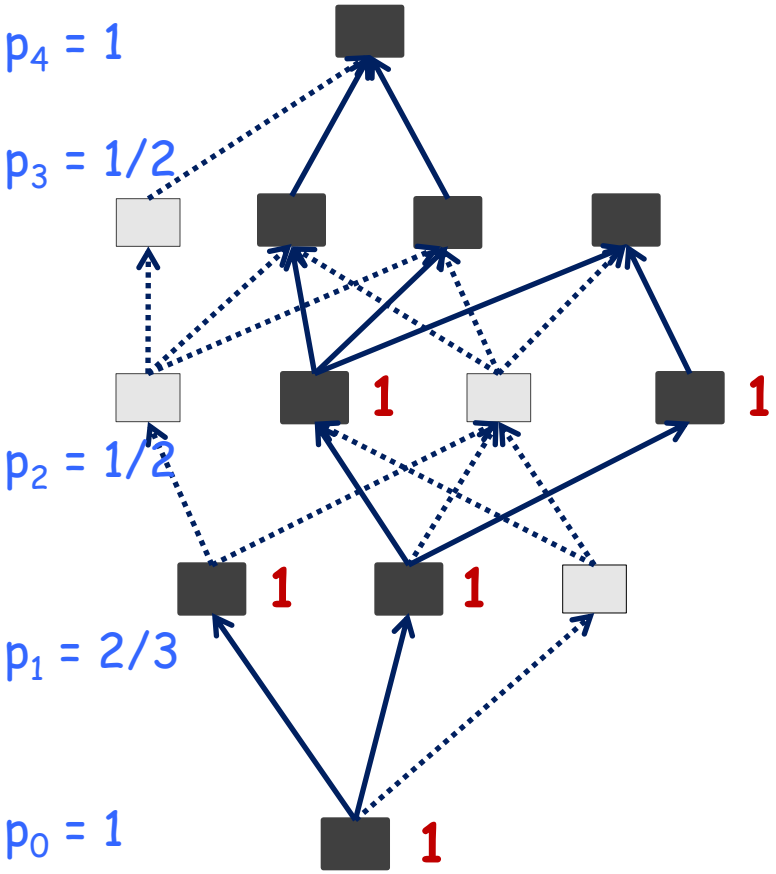
Estimating the number of source-sink paths in a graded DAG



$$Z = Ex[X] / P$$

$$P = \Pi_i p_i$$

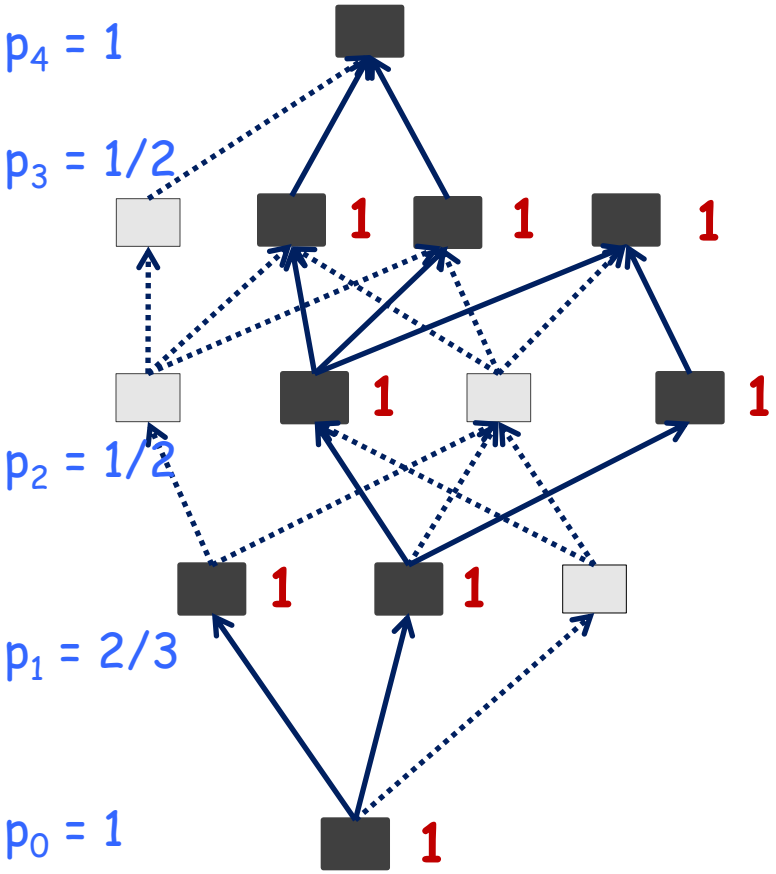
Estimating the number of source-sink paths in a graded DAG



$$Z = Ex[X] / P$$

$$P = \Pi_i p_i$$

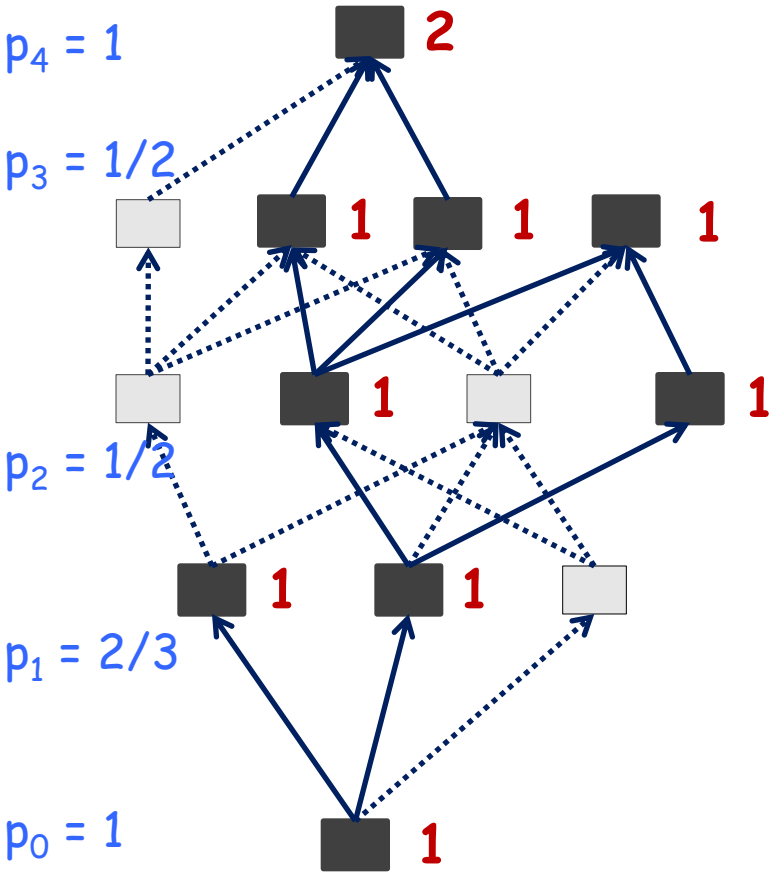
Estimating the number of source-sink paths in a graded DAG



$$Z = Ex[X] / P$$

$$P = \Pi_i p_i$$

Estimating the number of source-sink paths in a graded DAG



$$Z = Ex[X] / P$$

$$P = \Pi_i p_i = 1/6$$

Estimate for Z: 2/(1/6) = 12

How good is this estimator Y = X/P?

Y is unbiased E[Y] = Z (the true number of paths)

How good is this estimator Y = X/P?

Y is unbiased E[Y] = Z (the true number of paths)

so far no good concentration results for Y

$$\Pr(|Y - Ex[Y]| < c\, Ex[Y]) < ???$$

How good is this estimator Y = X/P?

Y is unbiased E[Y] = Z (the true number of paths)

so far no good concentration results for Y

$$\Pr(|Y - Ex[Y]| < c\, Ex[Y]) < ???$$

$$\Pr(|Y - Ex[Y]| < c\, Ex[Y]) < M / c^2$$

path coupling coefficient $M = \left(\sum_{w,v} 1/p^{C(w,v)} \right) / Z^2 - 1$

$C(w,v)$ = # of common vertices of paths w and v

each $p_i = p$

50 random points

50 random points

27 hours computing time using
128 GB main memory

50 random points

27 hours computing time using
128 GB main memory

147019897942999105259582587551602 triangulations

1.470198… ee_32

SAARLAND UNIVERSITY
COMPUTER SCIENCE

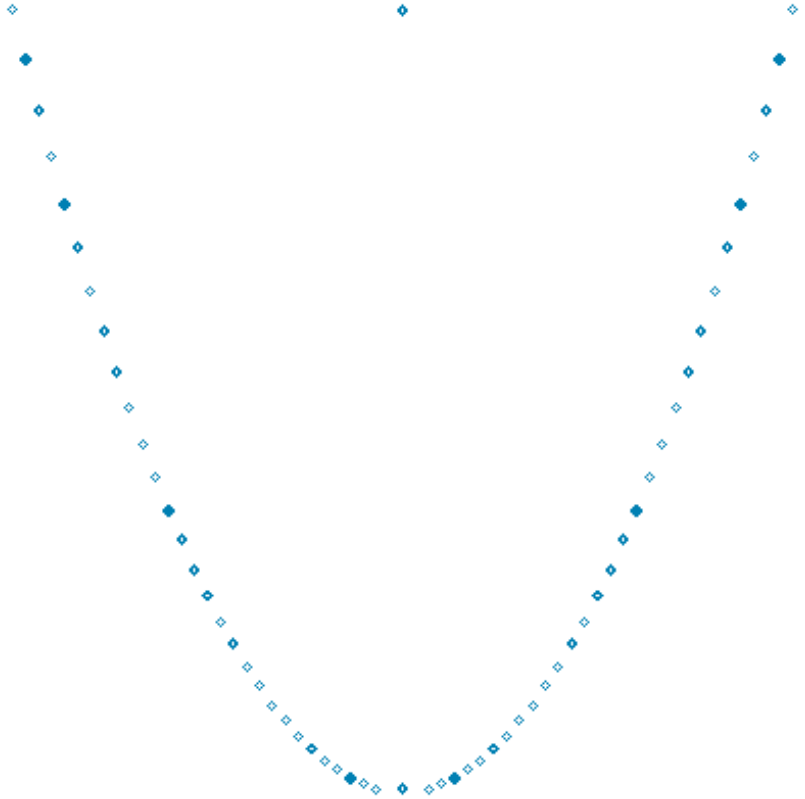
50 random points

27 hours computing time using
128 GB main memory

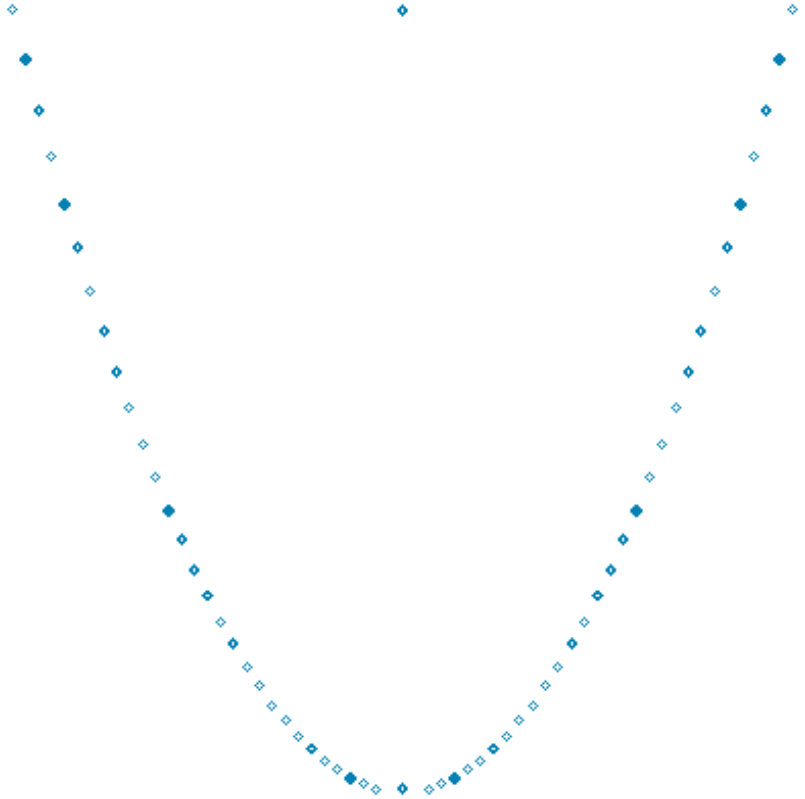
147019897942999105259582587551602 triangulations

1.470198… ee_32

SAARLAND UNIVERSITY
COMPUTER SCIENCE



„Double circle“ of 60 points



„Double circle“ of 60 points

$$\sum_{0 \le i \le h} \binom{h}{i} (-1)^{h-i} C(h+i-2)$$

triangulations, where h=n/2,

$$C(k) = \frac{1}{k+1}\binom{2k}{k}$$

146048501608765489640471405 50 triangulations

1.46048… ee_28

„Double chain“ of 60 points

„Double chain“ of 60 points

$$C(h-2)^2\binom{2h-2}{h-1}$$

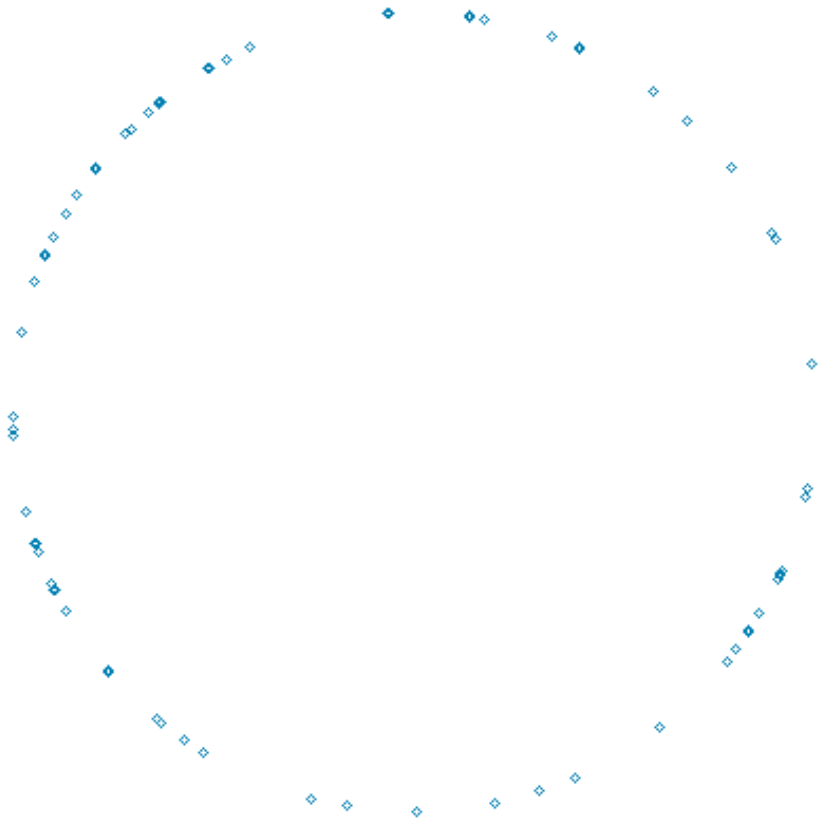
triangulations, where h=n/2 ,

$$C(k)=\frac{1}{k+1}\binom{2k}{k}$$

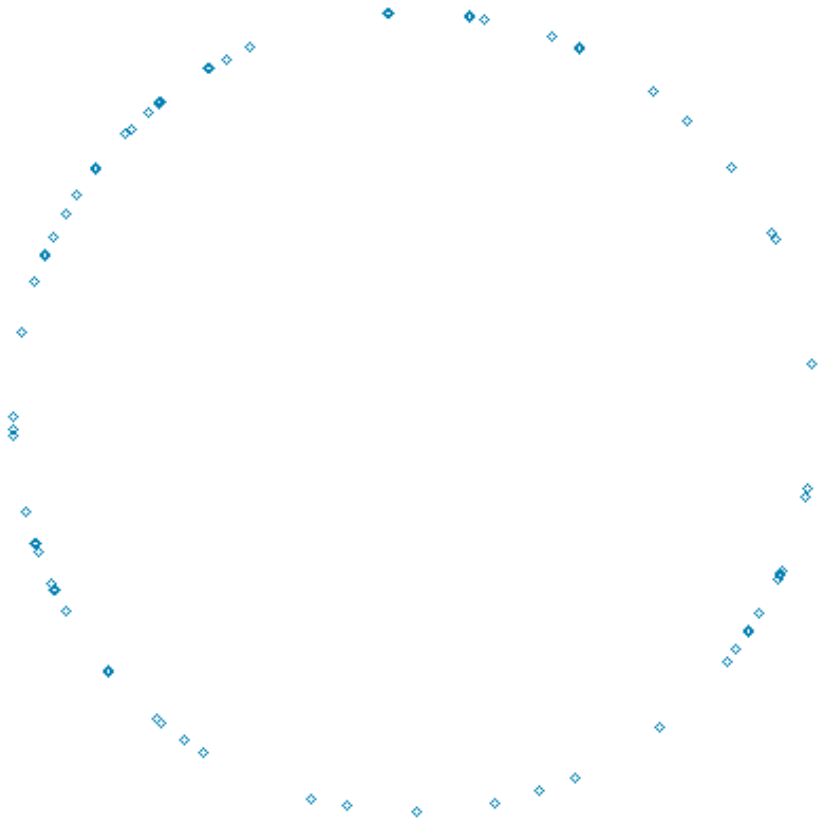
209156871987561535054674936116302001091878400 triang.s

2.09156… ee_45

SAARLAND UNIVERSITY
COMPUTER SCIENCE



60 points in convex position



60 points in convex position

C(n-2) triangulations, where

$$C(k) = \frac{1}{k+1}\binom{2k}{k}$$

1040884602891223040334983188120 80 triangulations

1.040884... ee_32

Problems

- Find useful bounds for the path coupling coefficient for the DAGs arising in triangulation counting for

  (i) general point sets
  (ii) special classes of point sets

Problems

- Find useful bounds for the path coupling coefficient for the DAGs arising in triangulation counting for

  (i) general point sets
  (ii) special classes of point sets

- Design a non-averaging strategy for obtaining a guaranteed approximator from the path counting estimator for

  (i) general point sets
  (ii) special classes of point sets

Problems

- Design efficient methods or heuristics recognizing dead end nodes in the DAGs arising in triangulation counting.

  (recognize marked sweep chains that cannot be completed to a triangulation)

Problems

- Design efficient methods or heuristics recognizing dead end nodes in the DAGs arising in triangulation counting.

  (recognize marked sweep chains that cannot be completed to a triangulation)

- Find other counting problems for which DAG path estimation can be profitably employed. (matching?)

Problems

- Design efficient methods or heuristics recognizing dead end nodes in the DAGs arising in triangulation counting.

  (recognize marked sweep chains that cannot be completed to a triangulation)

- Find other counting problems for which DAG path estimation can be profitably employed. (matching?)

- Where else has DAG path estimation been used, and under what name?

Easy Generalizations

Count triangulations of S that are constrained to contain certain edges.

Truly uniform random generation of triangulations of a given set S.

Fast unbiased estimation of tr(S).

Not so easy Generalizations

Counting non-crossing matchings, plane geometric graphs, spanning cycles, ….

Manuel Wettstein, MSc Thesis, ETH, April 2013
SOCG 2014

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Not so easy Generalizations

Counting non-crossing matchings, plane geometric graphs, spanning cycles, ….

Manuel Wettstein, MSc Thesis, ETH, April 2013
SOCG 2014

Talk this Friday at 10:30


SAARLAND
UNIVERSITY
COMPUTER SCIENCE