Interactive Machine Learning

Navigation

Cédric Buche

http://www.enib.fr/ ~buche

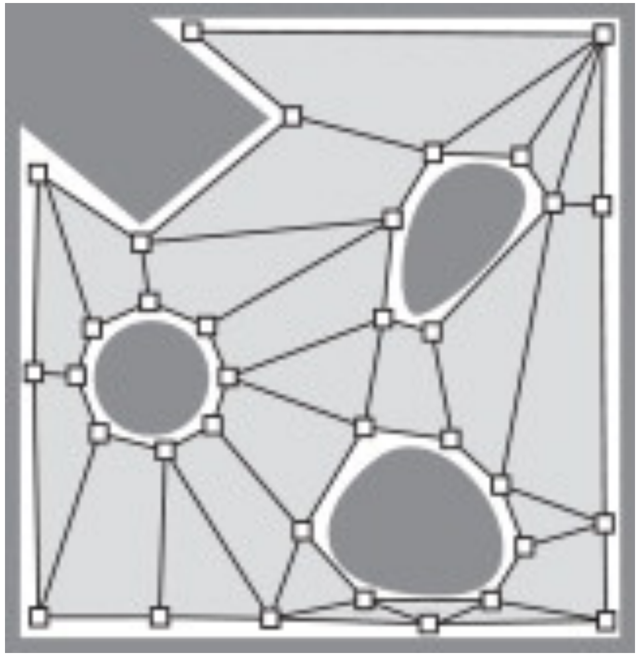
Goal

- Navigation : determine paths to go from one point to an other.

Summary

- **Classical approaches**
- **GNG**
- **SGNG**

Mesh



A simple environment (obstacles are in dark grey) represented by a mesh. The avatar can navigate in the zone defined by the mesh (in grey) because it knows there are no obstacles in this zone. Different algorithms can be used to find optimal paths.

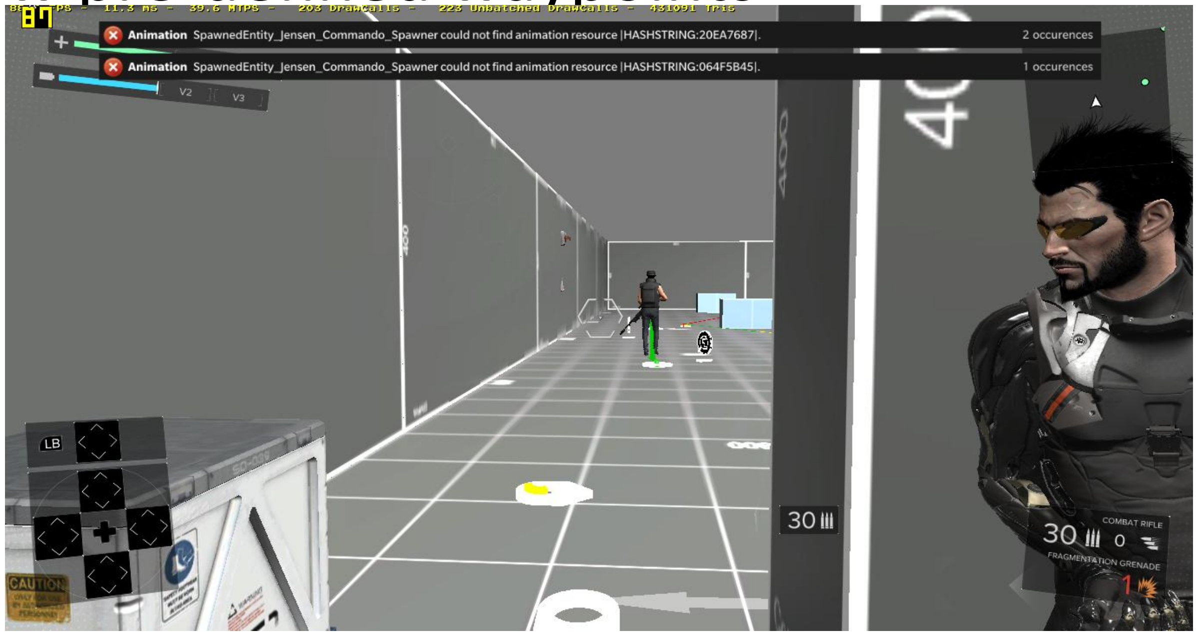
- requires an algorithm to find the optimal path between two points
- a path which may not be natural or believable

Graph

A simple environment (obstacles are in grey) represented by a graph. Nodes are represented by circles and edges by black lines. An avatar can move from one node to another only if the nodes are connected by an edge. Usually, an A* is used to find the path between two nodes.

Video games

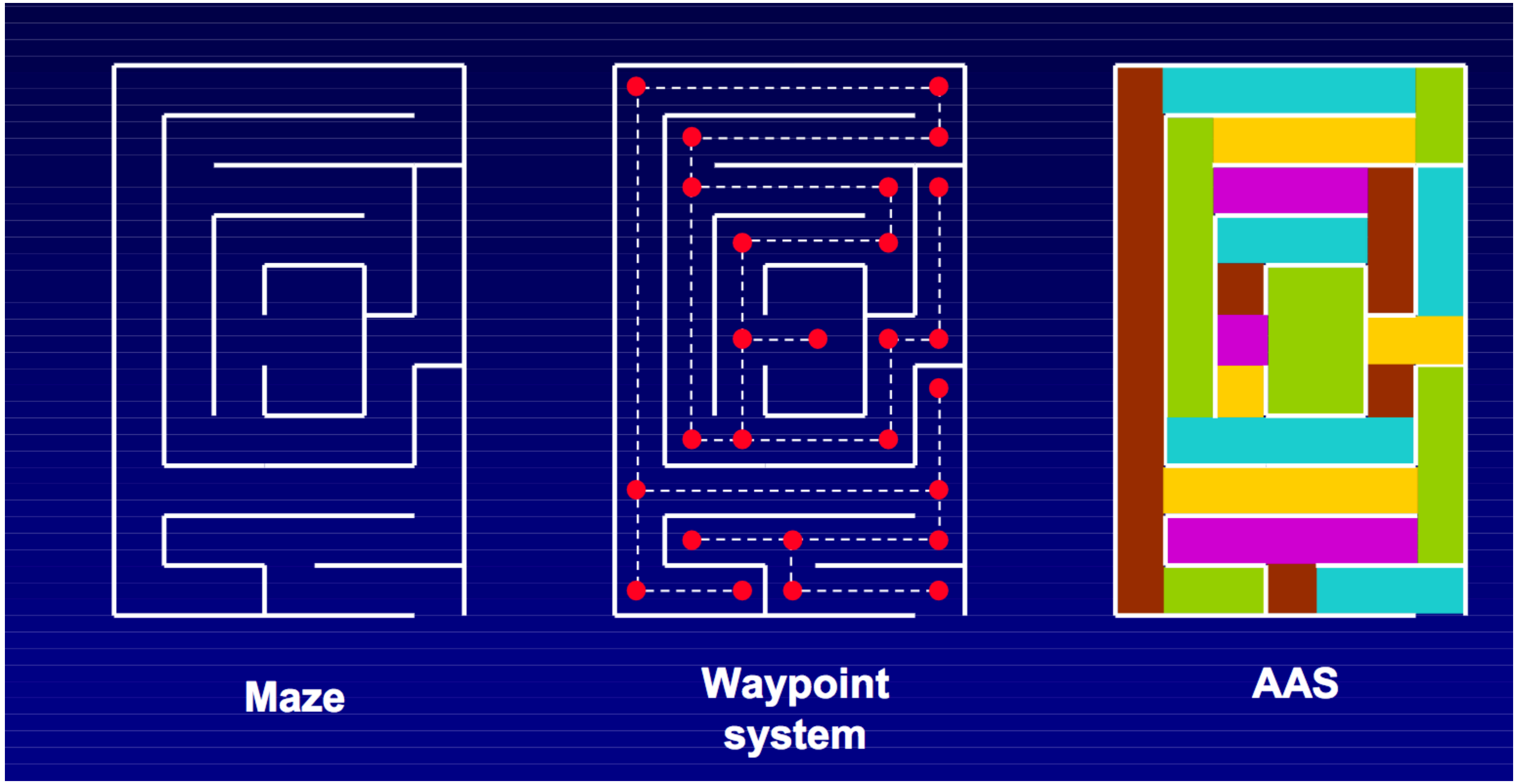
- Classical bots are designed to follow pre-defined waypoints determined by the map designer.



- These bots need to have a waypoint file for each map, or a pathnode system embedded in the map.

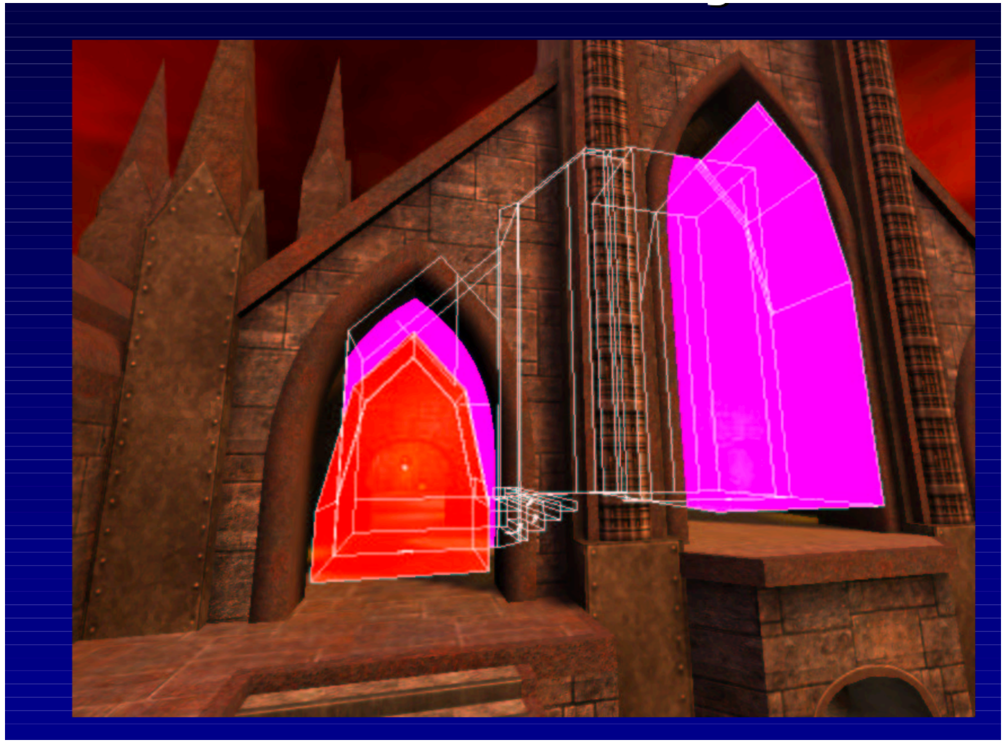
Quake 3 Arena bots

- use an area awareness system file to move around the map



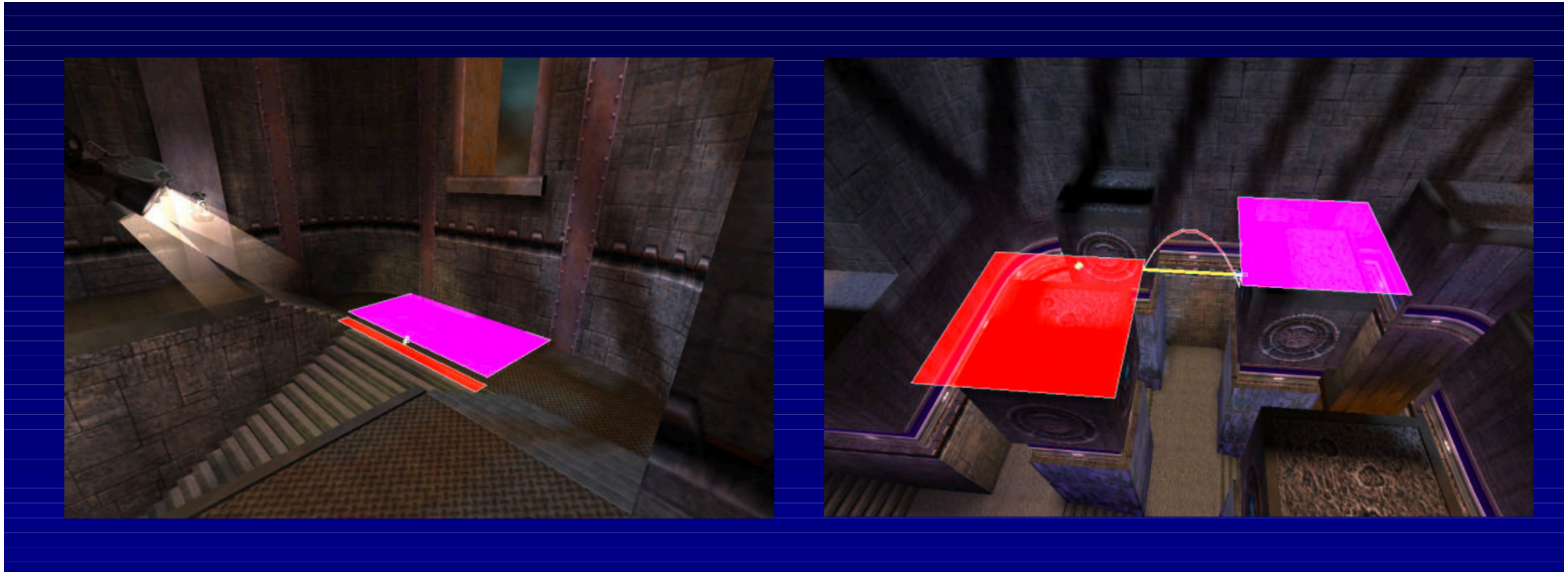
Quake 3 Arena bots

- use an area awareness system file to move around the map



Quake 3 Arena bots

- use an area awareness system file to move around the map



Counter-Strike bots

- use a waypoint file

Unreal Tournament's series bots

- use a pathnode system embedded in the map to navigate

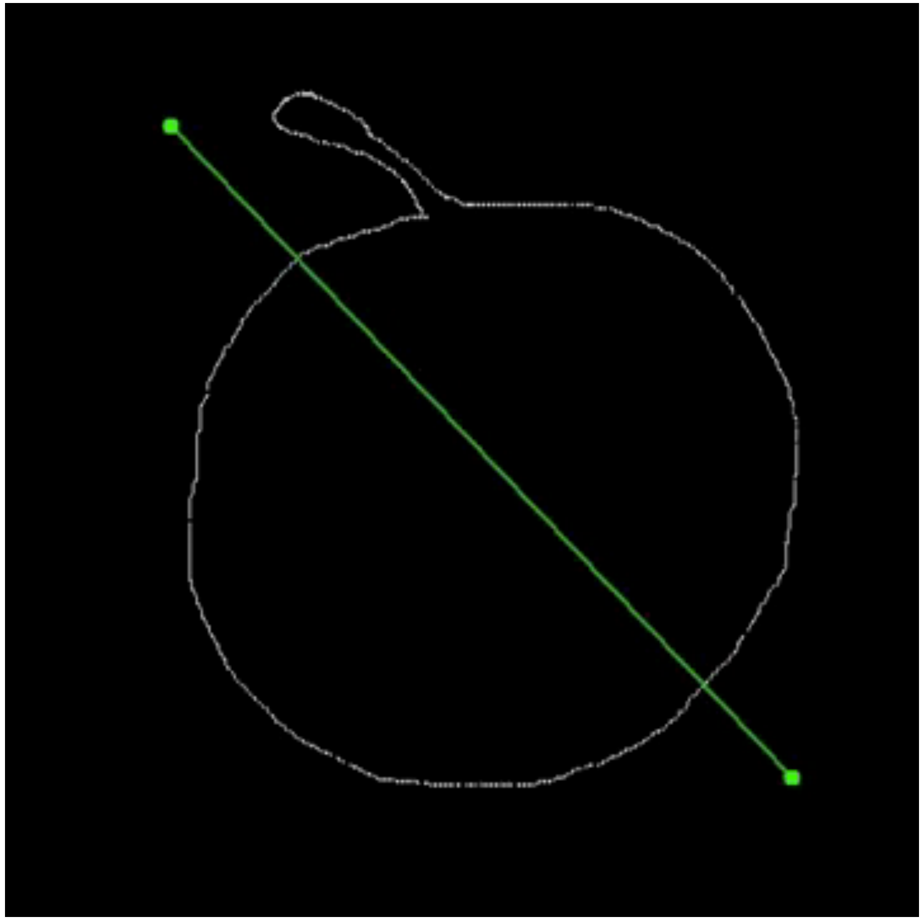
Valve bots

- To support the many community-created maps, some games include an automatic mesh generation system
- The first time users attempt to play a custom map with bots, the generation system will build a navigation file for that map. Starting at a player spawn point, walkable space is sampled by “flood-filling” outwards from that spot, searching for adjacent walkable points.
- Finally, dynamic bots are able to dynamically learn levels and maps as they play. RealBot, for *Counter-Strike*, is an example of this. However, this learning is not guided by human behavior.
- Navigation points obtained will therefore not produce believable behavior. The paths the bots use to go from one point in the environment to another do not resemble those human players would take. This problem comes not from the decision-making process itself, but from the representation of the environment it uses. Indeed, the bots use navigation points in the environment which may not accurately or naturally represent how players use the same environment.

Summary

- **Classical approaches**
- **GNG**
- **SGNG**

GNG



GNG

- Error = how well the node represents its surroundings
- The fewer errors a node has, the better it represents its surroundings

GNG

while Number of nodes $\leq N_{max}$ **do**

Get input position (4 (a))

Pick the closest (n_1) and the second closest nodes (n_2) (4 (b))

Create edge between n_1 and n_2 (4 (c)).

If an edge already existed, reset its age to 0.

Increase the error of n_1 (4 (d))

Move n_1 and its neighbors toward the input (4 (e))

Increase the age of all the edges emanating from n_1 by 1 (4 (f))

Delete edges exceeding a certain age (4 (g))

if Iteration number is a multiple of η **then**

Find the maximum error node n_{max}

Find the maximum error node n_{max2} among the neighbor of n_{max} (4 (h))

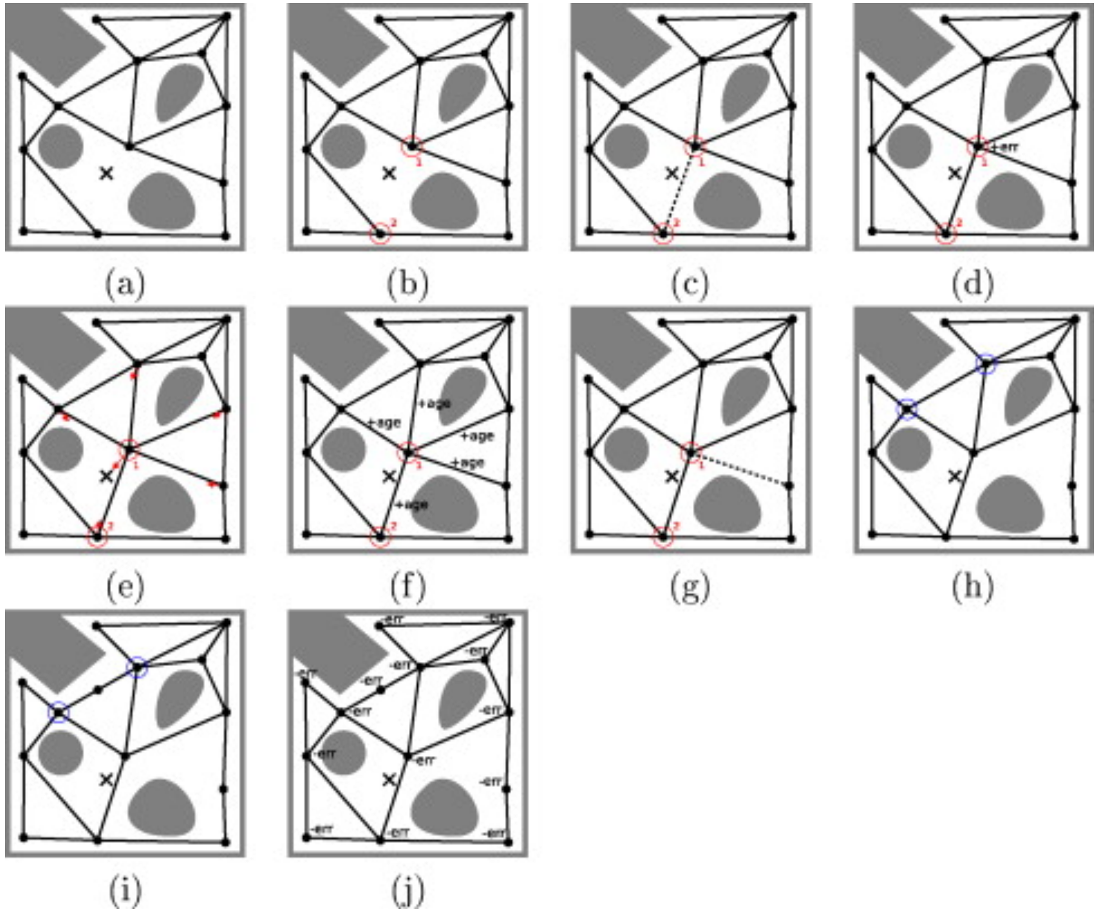
Insert node between n_{max} and n_{max2} (4 (i))

Decrease the error of n_{max} and n_{max2}

end if

Decrease each node's error by a small amount (4 (j))

end while



GNG

- Fails to handle temporal series
- Grow up over the time

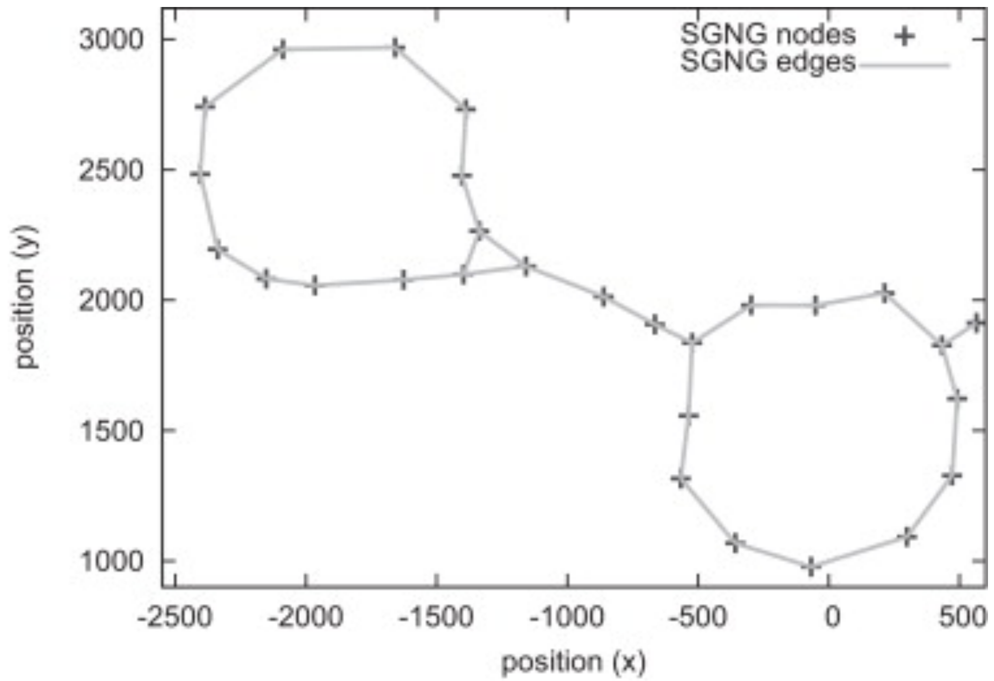
Summary

- **Classical approaches**
- **GNG**
- **SGNG**

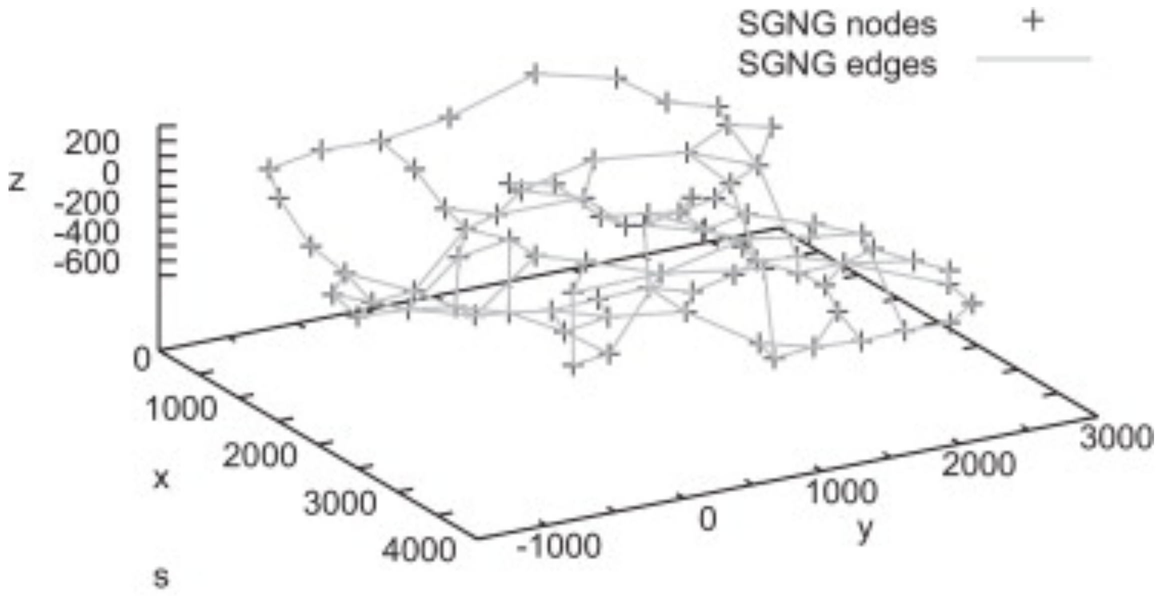
SGNG

```
nodes ← {}
edges ← {}
while teacher plays do
  (x,y,z) ← teacher's position
  if |nodes| = 0 or 1 then
    nodes ← nodes ∪ {(x,y,z,error=0)}
  end if
  if |nodes| = 2 then
    edges ← {(nodes,age=0)}
  end if
  n_1 ← closest((x,y,z),nodes)
  n_2 ← secondClosest((x,y,z),nodes)
  edge ← edges ∪ {{n_1,n_2},age=0)}

  n_1.error+=||(x,y,z)-n_1||
  Attract n_1 toward (x,y,z)
  ∀ edge ∈ edgesFrom(n_1), edge.age++
  Delete edges older than \overline{Age}
  Attract neighbors(n_1) toward (x,y,z)
  ∀ node ∈ nodes, node.error-=Err

  if n_1.error > \overline{Err} then
    maxErrNei ← maxErrorNeighbour(n_1)
    newNode ← between(n_1,maxErrNei)
    n_1.error/=2
    maxErrNei.error/=2
    newError ← n_1.error+maxErrNei.error
    nodes ← nodes ∪ {(newNode,newError)}
  end if
end while
```

SGNG

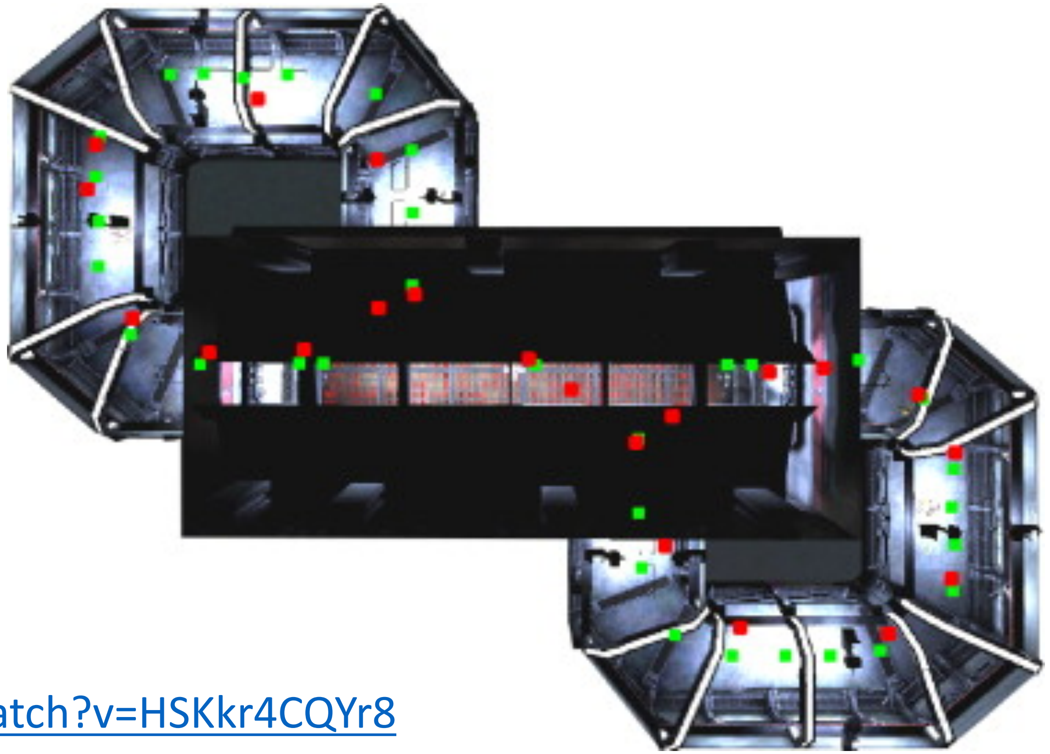


SGNG nodes
SGNG edges
3000
2500
2000
1500
1000
position (y)
-2500
-2000
-1500
-1000
-500
0
500
position (x)



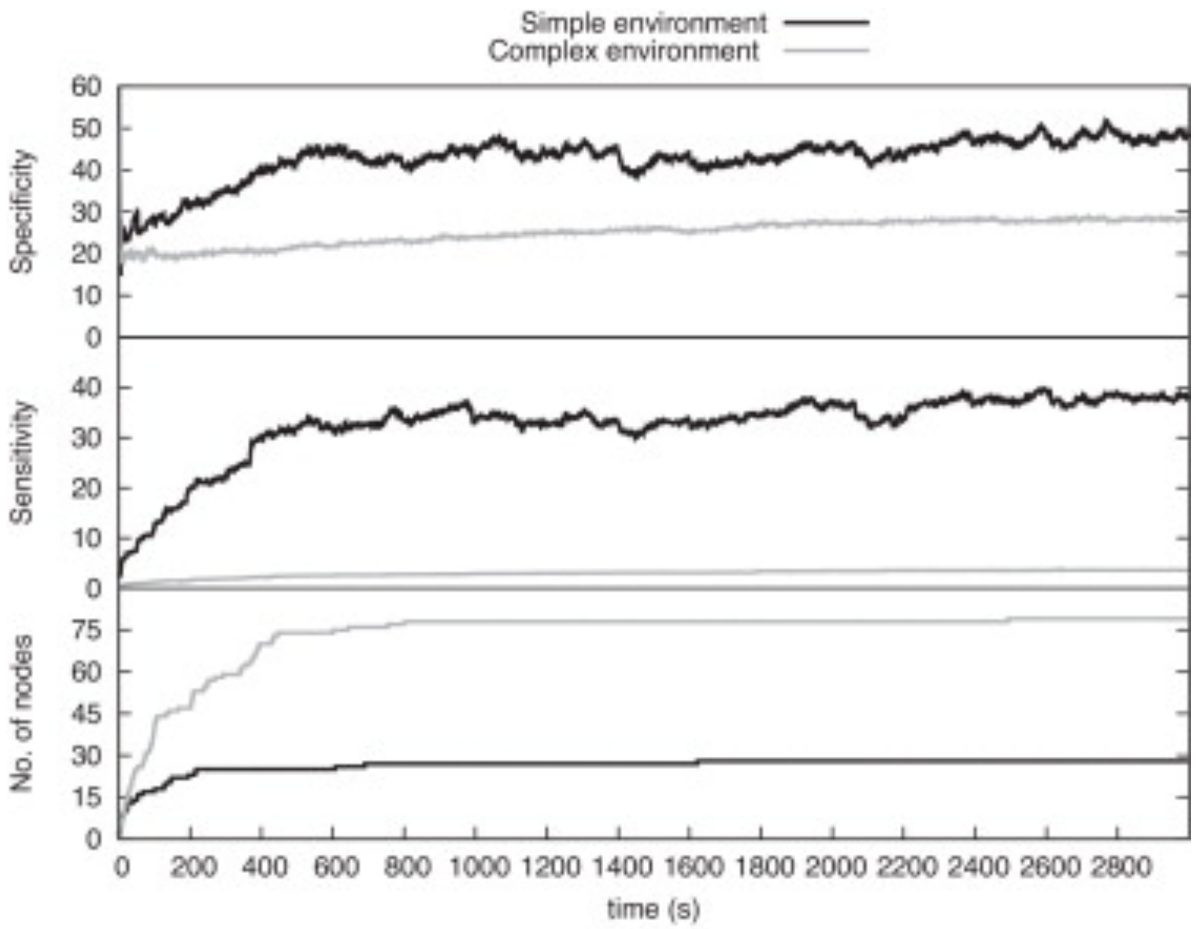
SGNG nodes
SGNG edges
200
0
-200
-400
-600
z
0
1000
2000
3000
4000
x
s
-1000
0
1000
2000
3000
y

SGNG



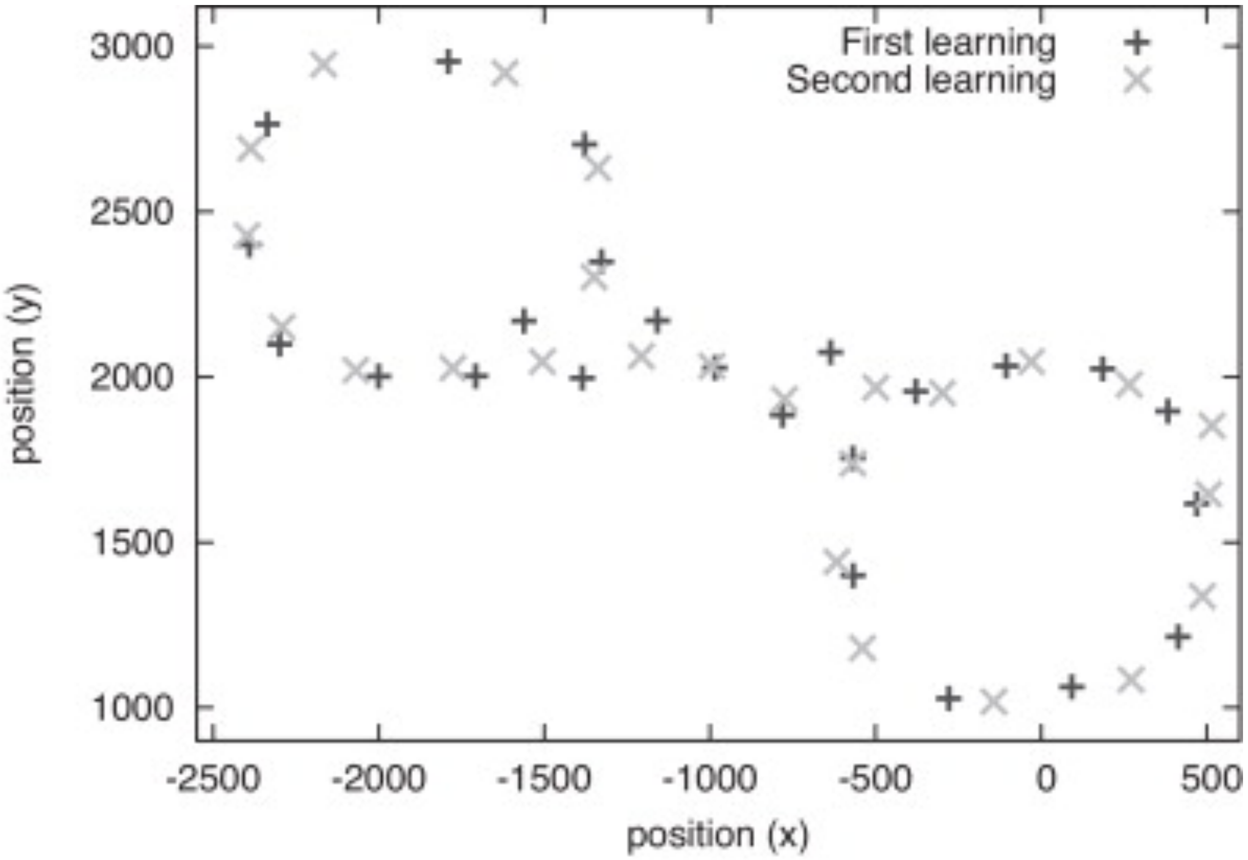
https://www.youtube.com/watch?v=HSKkr4CQYr8

SGNG

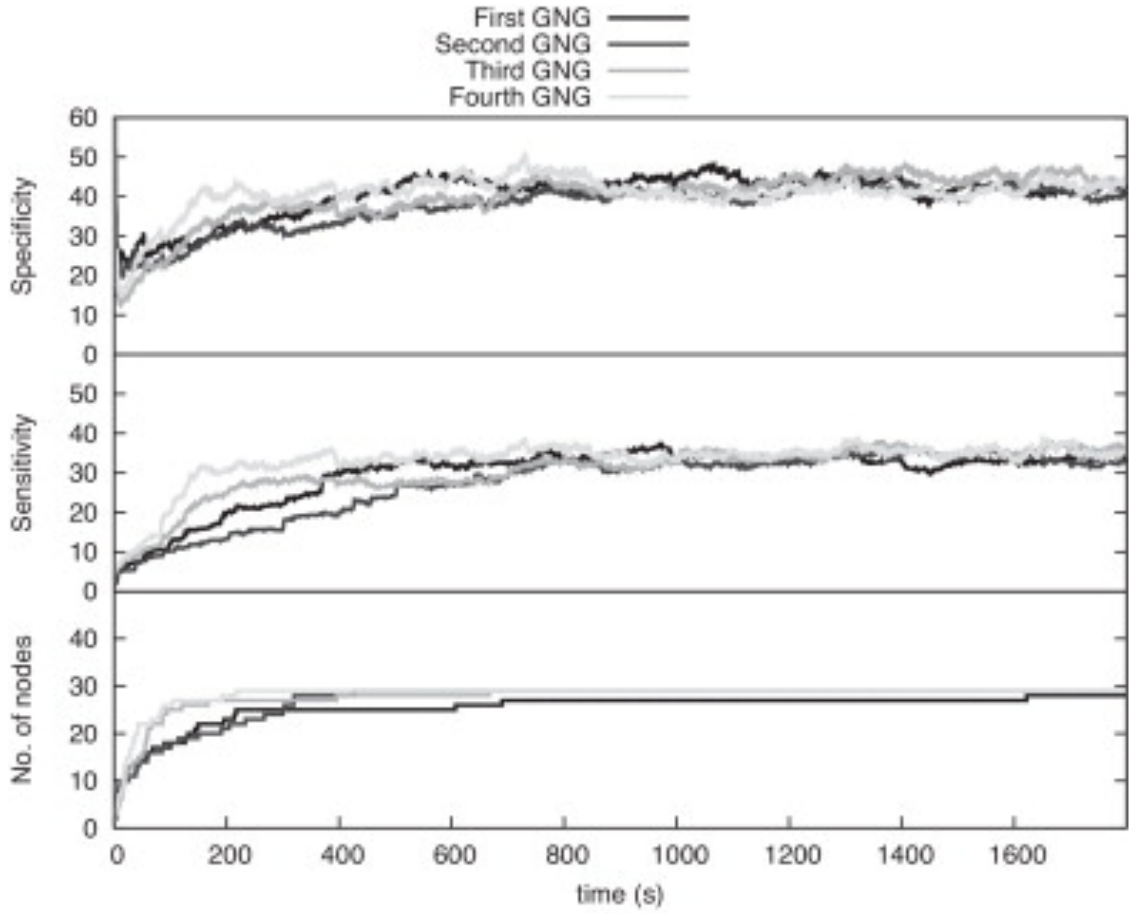


Simple environment
Complex environment
Specificity
Sensitivity
No. of nodes
0
10
20
30
40
50
60
0
10
20
30
40
0
15
30
45
60
75
0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600 2800
time (s)

SGNG

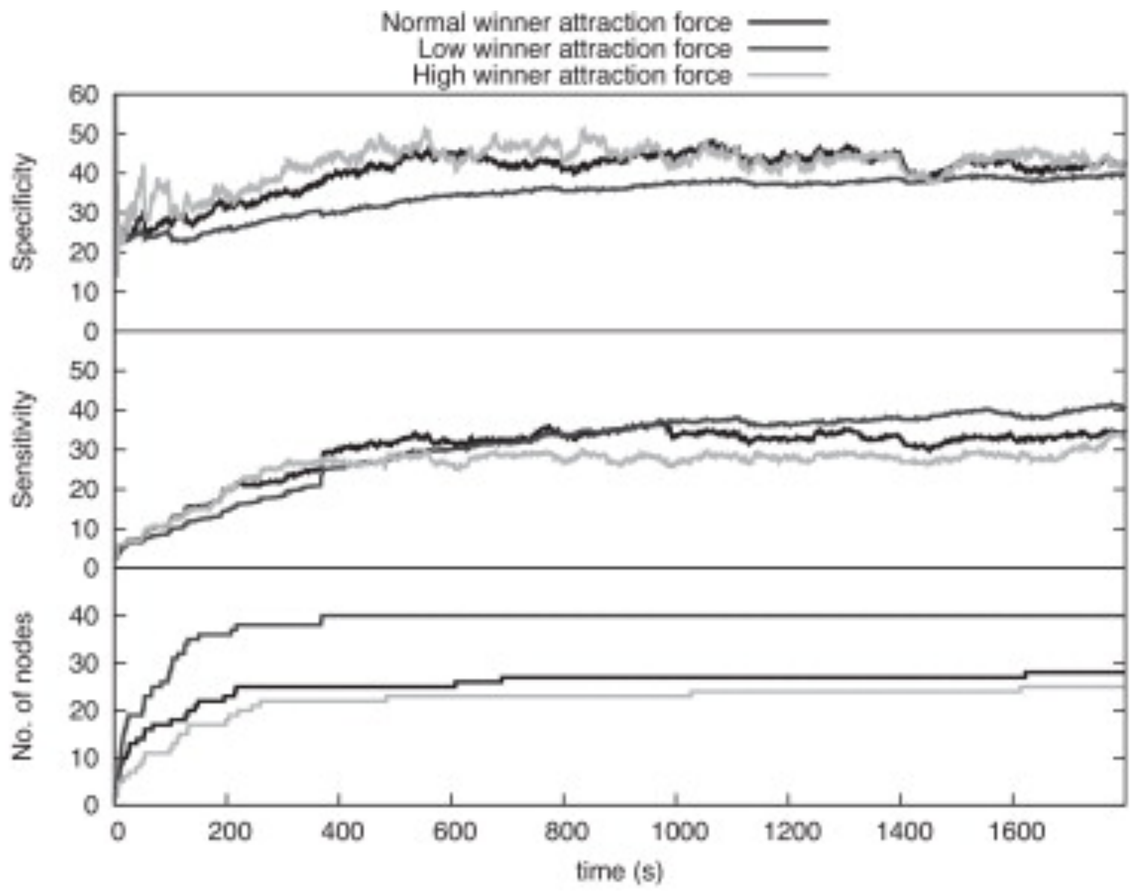

First learning
Second learning
position (y)
position (x)
3000
2500
2000
1500
1000
-2500
-2000
-1500
-1000
-500
0
500

SGNG



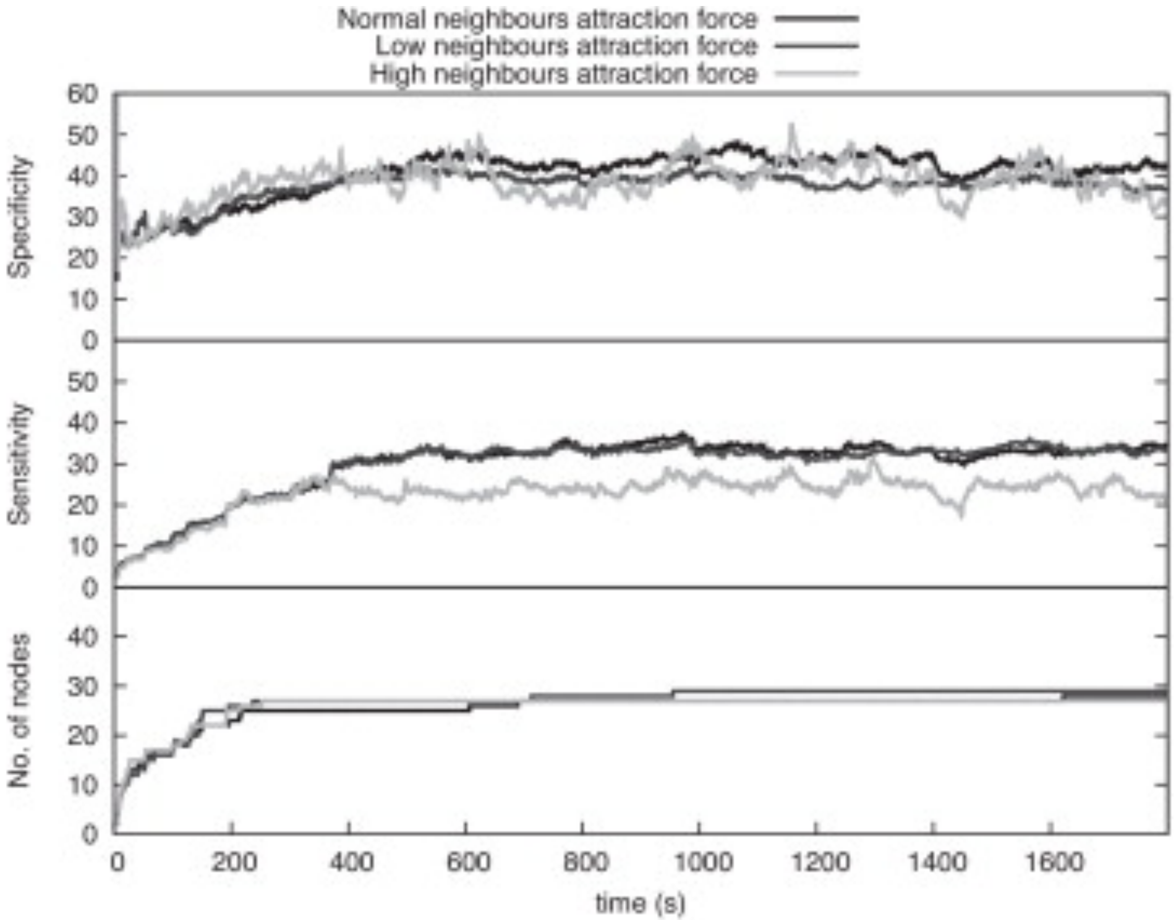
First GNG
Second GNG
Third GNG
Fourth GNG
Specificity
Sensitivity
No. of nodes
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
1400
1600
time (s)

SGNG



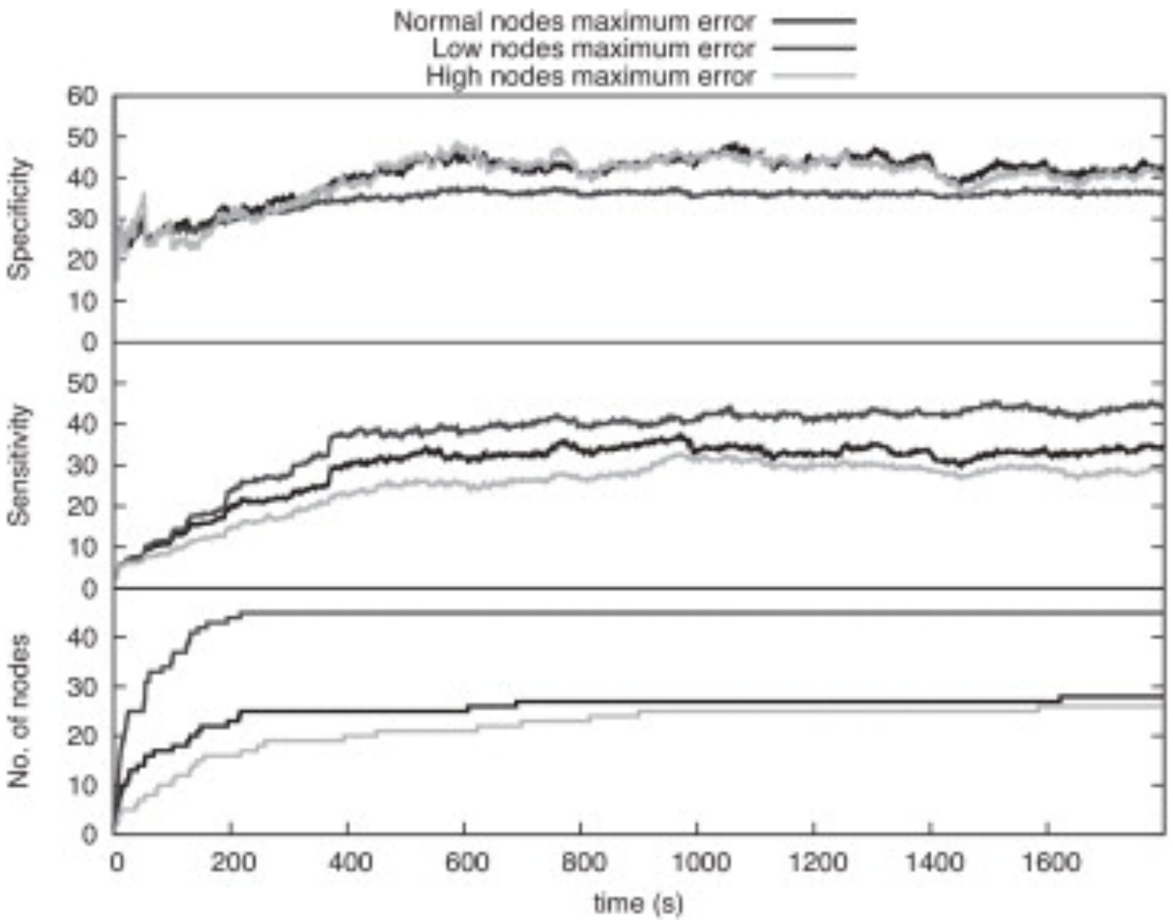
Normal winner attraction force
Low winner attraction force
High winner attraction force
Specificity
Sensitivity
No. of nodes
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
1400
1600
time (s)

SGNG



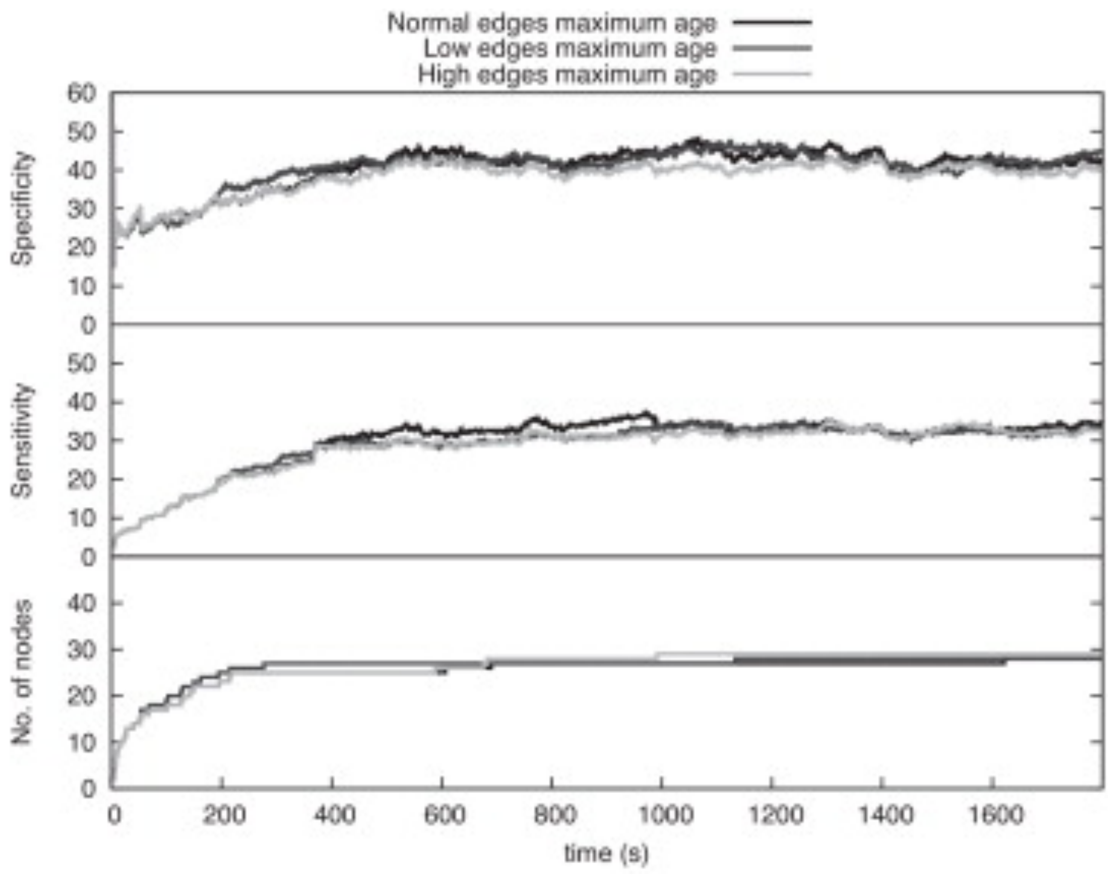
Normal neighbours attraction force
Low neighbours attraction force
High neighbours attraction force
Specificity
Sensitivity
No. of nodes
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
1400
1600
time (s)

SGNG

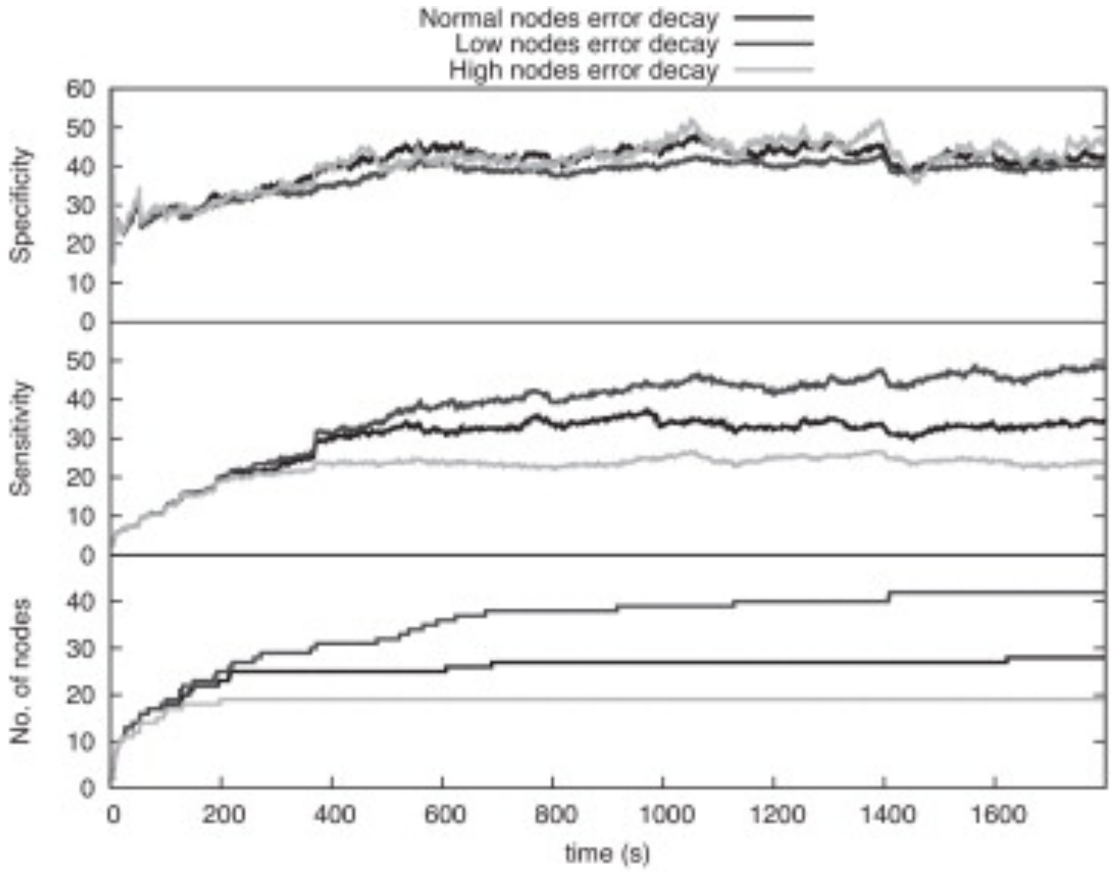


Normal nodes maximum error
Low nodes maximum error
High nodes maximum error
Specificity
Sensitivity
No. of nodes
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
1400
1600
time (s)

SGNG

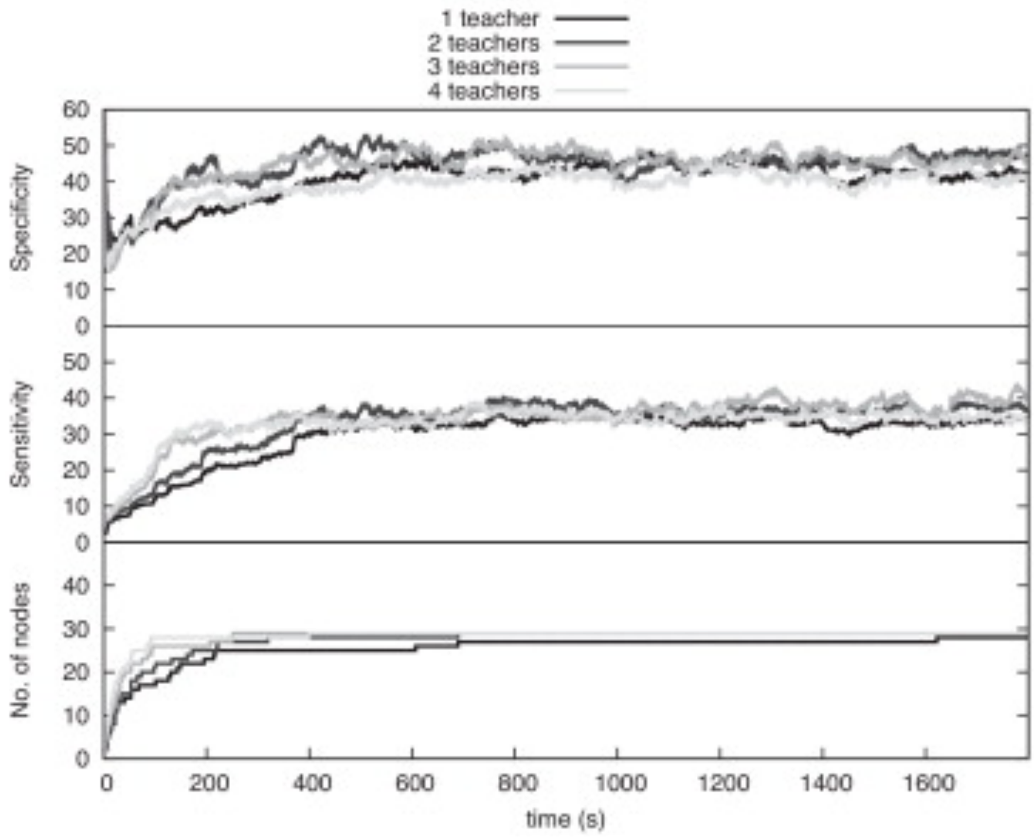

Normal edges maximum age
Low edges maximum age
High edges maximum age
Specificity
Sensitivity
No. of nodes
time (s)
0
200
400
600
800
1000
1200
1400
1600
10
20
30
40
50
60

SGNG

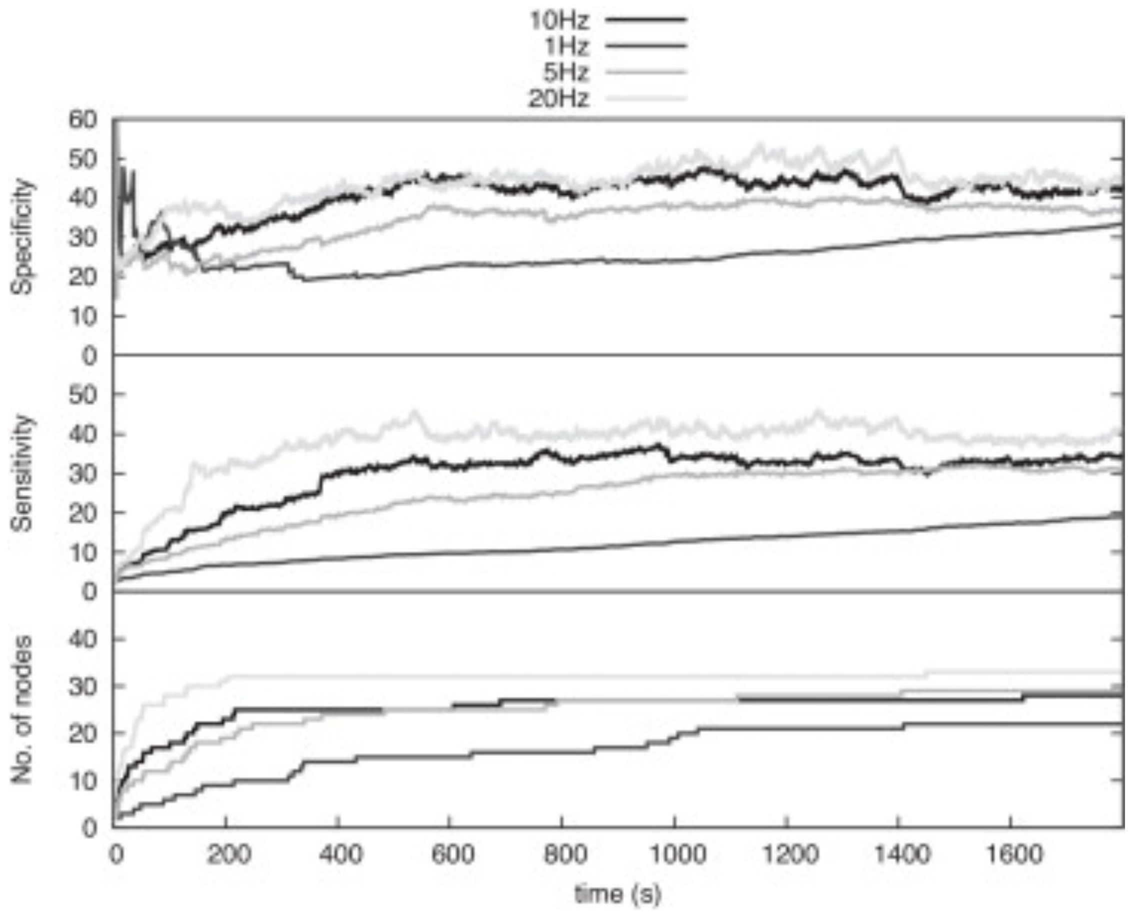


Normal nodes error decay
Low nodes error decay
High nodes error decay
Specificity
Sensitivity
No. of nodes
time (s)
0
200
400
600
800
1000
1200
1400
1600
10
20
30
40
50
60

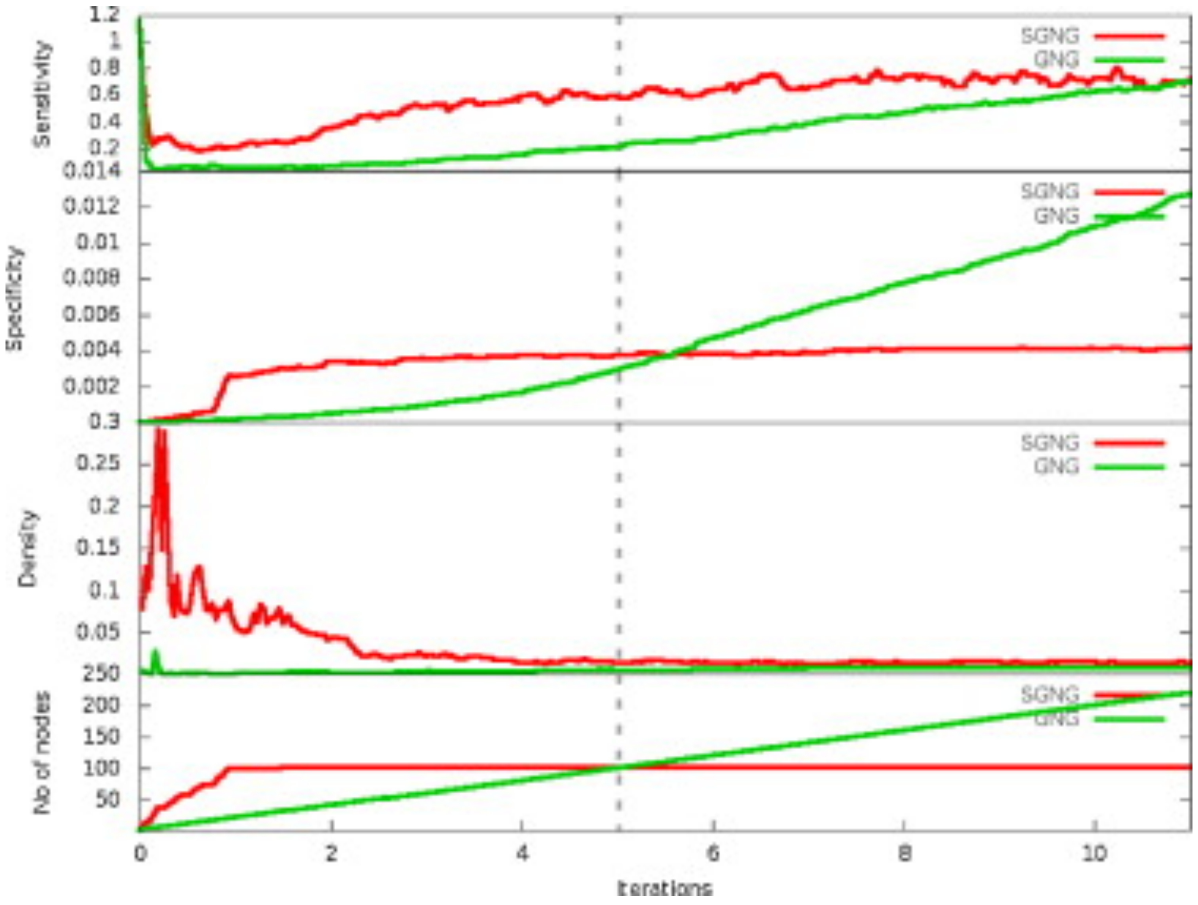
SGNG


1 teacher
2 teachers
3 teachers
4 teachers
Specificity
Sensitivity
No. of nodes
0
200
400
600
800
1000
1200
1400
1600
time (s)

SGNG

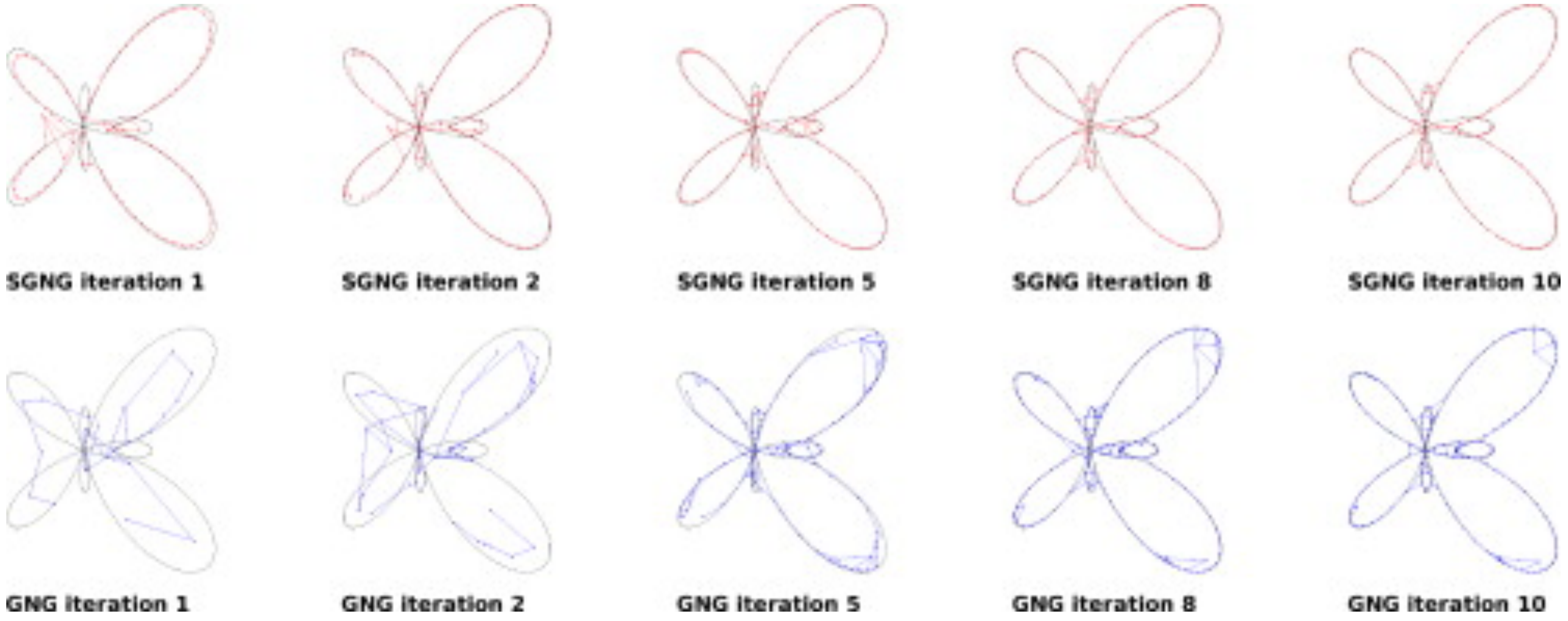

10Hz
1Hz
5Hz
20Hz
Specificity
Sensitivity
No. of nodes
60
50
40
30
20
10
0
time (s)
0
200
400
600
800
1000
1200
1400
1600

SGNG

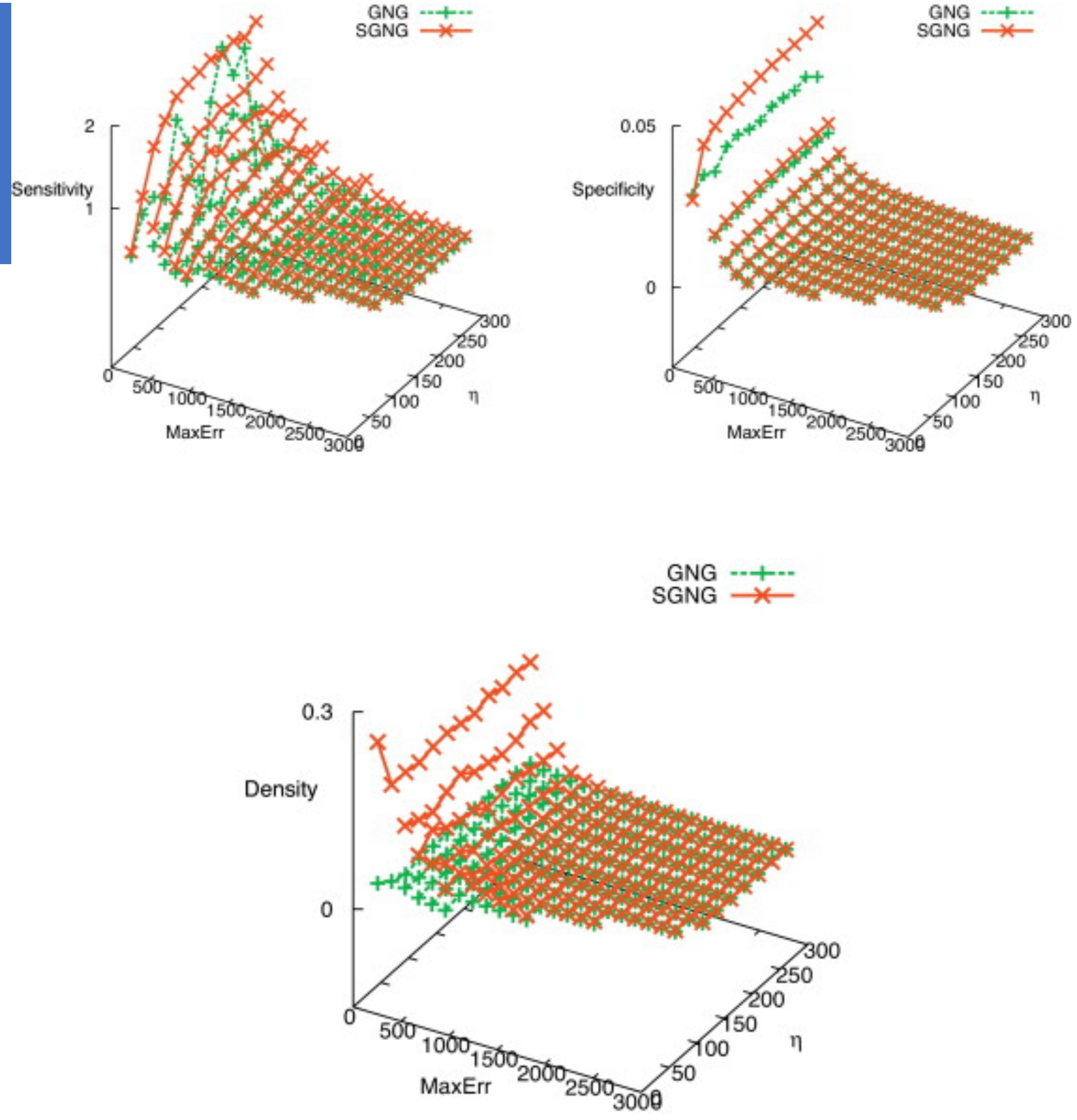


Sensitivity
Specificity
Density
Nb of nodes
SGNG
GNG
Iterations

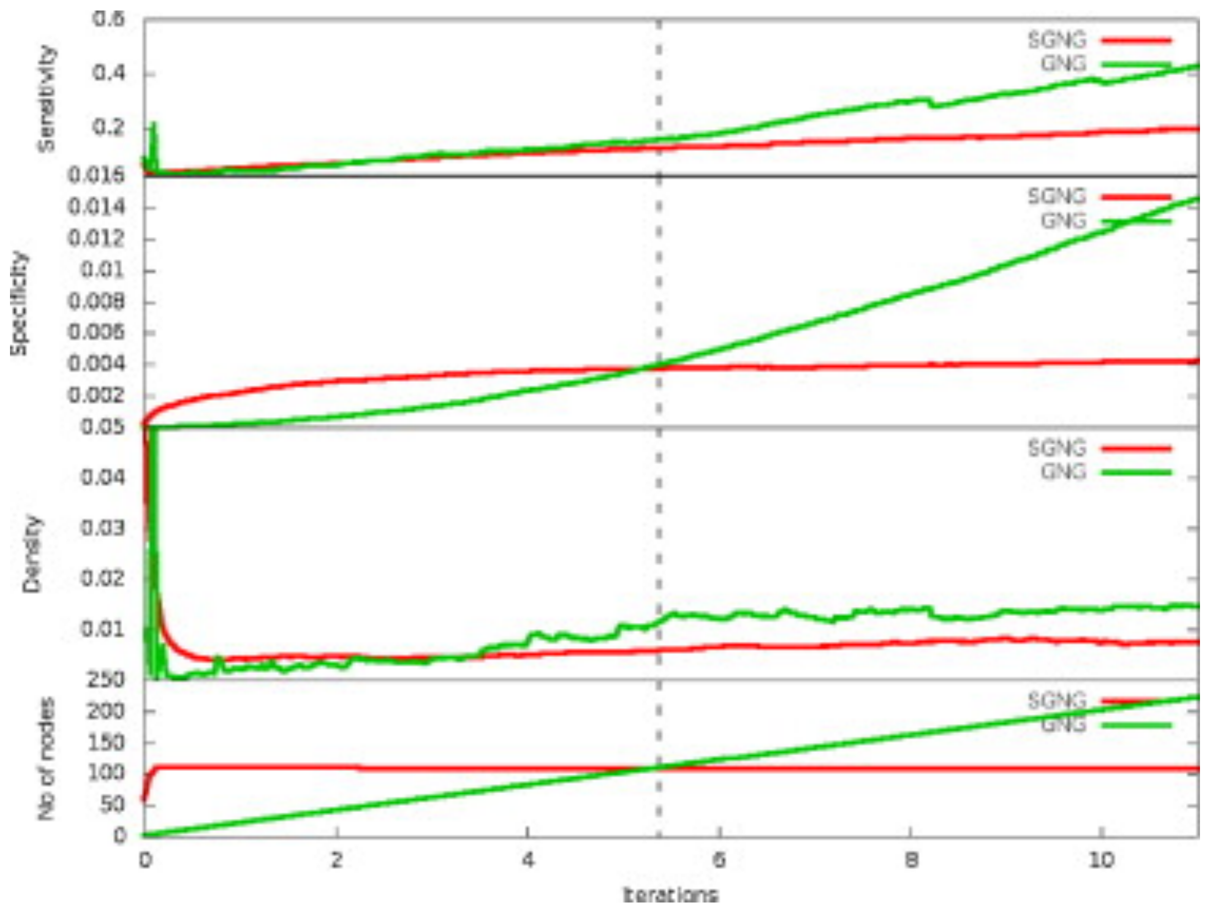
SGNG


SGNG iteration 1
SGNG iteration 2
SGNG iteration 5
SGNG iteration 8
SGNG iteration 10
GNG iteration 1
GNG iteration 2
GNG iteration 5
GNG iteration 8
GNG iteration 10

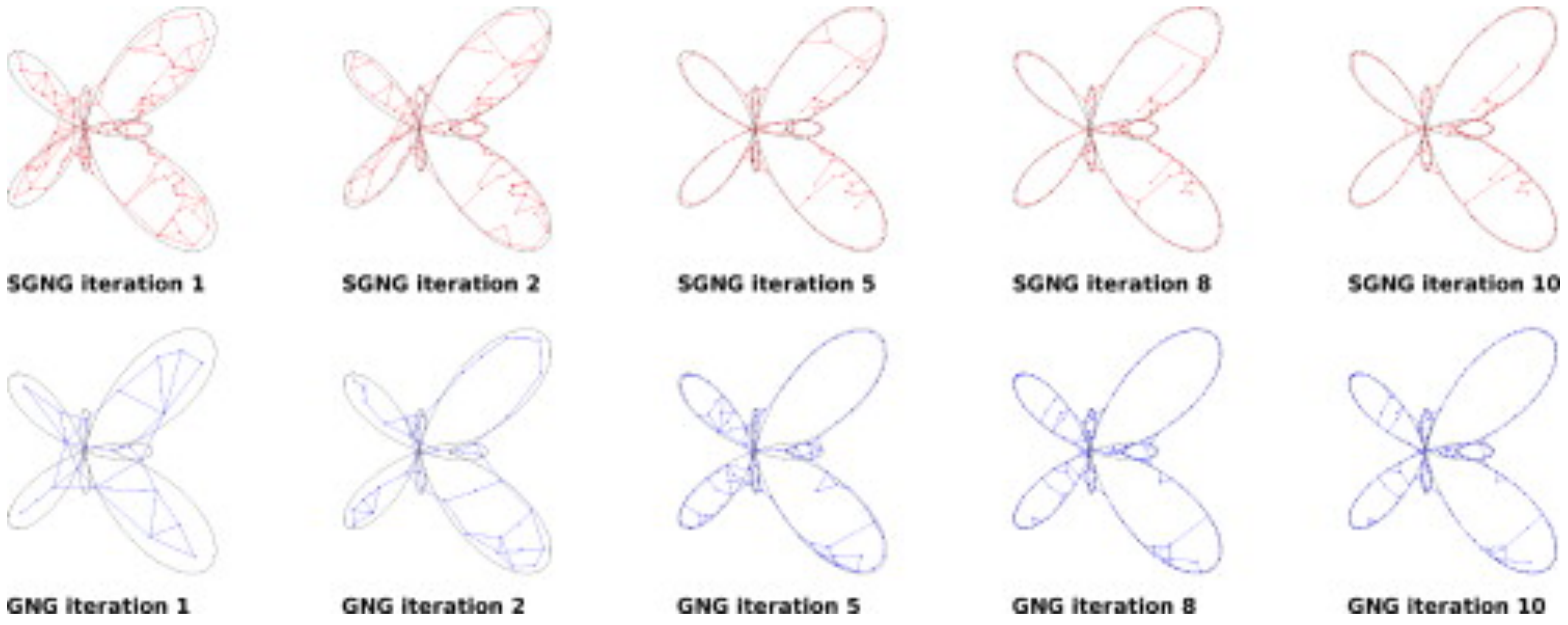
SGNG


GNG
SGNG
Sensitivity
2
1
0
500
1000
1500
2000
2500
3000
MaxErr
300
250
200
150
100
50
η
Specificity
0.05
0
Density
0.3

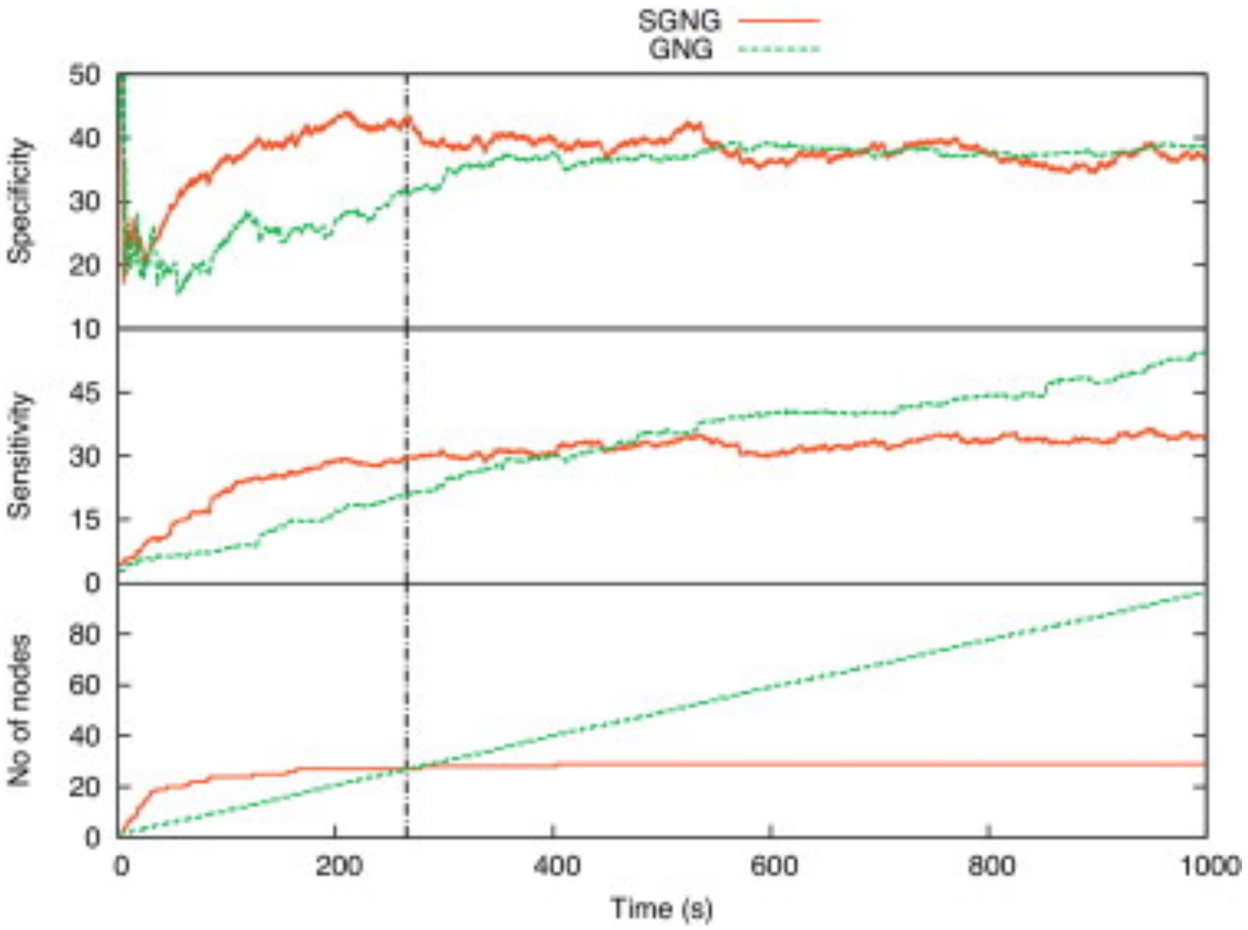
SGNG


Sensitivity
Specificity
Density
Nb of nodes
SGNG
GNG
0.6
0.4
0.2
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0.05
0.04
0.03
0.02
0.01
250
200
150
100
50
0
0
2
4
6
8
10
Iterations

SGNG


SGNG iteration 1
SGNG iteration 2
SGNG iteration 5
SGNG iteration 8
SGNG iteration 10
GNG iteration 1
GNG iteration 2
GNG iteration 5
GNG iteration 8
GNG iteration 10

SGNG


SGNG
GNG
Specificity
Sensitivity
No of nodes
50
40
30
20
10
45
30
15
0
80
60
40
20
0
0
200
400
600
800
1000
Time (s)

SGNG

F. Tence, L. Gaubert, J. Soler, P. De Loor, and C. Buche.
Stable Growing Neural Gas: a Topology Learning Algorithm based on Player Tracking in Video Games.
Applied Soft Computing (ASC),
13(10):4174-4184, 2013.

Interactive Machine Learning

Navigation

Cédric Buche

http://www.enib.fr/ ~buche