Hierarchical organization versus self-organization

Eva Busseniers

UGent

10 juni 2011

Table of contents

Hierarchy

- Structure so that for some it is easy to oppress/force their views for their own profit, while for most, it is the easiest to follow

Hierarchy

- Structure so that for some it is easy to oppress/force their views for their own profit, while for most, it is the easiest to follow
- Unequality

Hierarchy

- Structure so that for some it is easy to oppress/force their views for their own profit, while for most, it is the easiest to follow
- Unequality
- Order: some are "higher" then others

Hierarchy

- Structure so that for some it is easy to oppress/force their views for their own profit, while for most, it is the easiest to follow
- Unequality
- Order: some are "higher" then others
- Unequal relations: one has power over the other (local)

Hierachy vs self-organization

- Organization = structure with function

Hierachy vs self-organization

- Organization = structure with function
- Hierarchical organization vs self-organization

Hierachy vs self-organization

- Organization = structure with function
- Hierarchical organization vs self-organization
- Two ways to approach it:
  - difference on structure
  - different way in achieving goal

Hierachy vs self-organization

- Organization = structure with function
- Hierarchical organization vs self-organization
- Two ways to approach it:
  - difference on structure
  - different way in achieving goal
- see how the two agree

Network

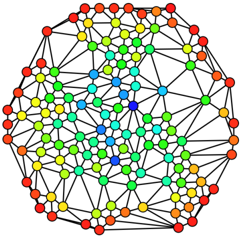
- Structure $\rightarrow$ by network/graph

Network

- Structure $\rightarrow$ by network/graph
- Graph $=$ 'nodes' connected by 'edges'

Network

- Structure $\rightarrow$ by network/graph
- Graph = 'nodes' connected by 'edges'
- $G(V, E)$:
  - V = set of vertices/nodes
  - E set of edges (pair of nodes)



Degree

- Degree of node = number of neighbours

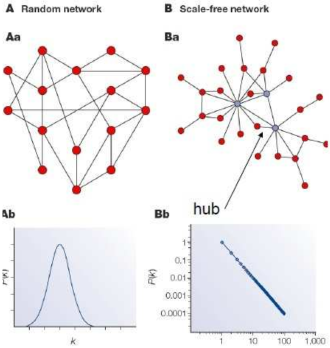
Degree

- Degree of node = number of neighbours
- Hierarchical= a lot of difference in degree?

Degree

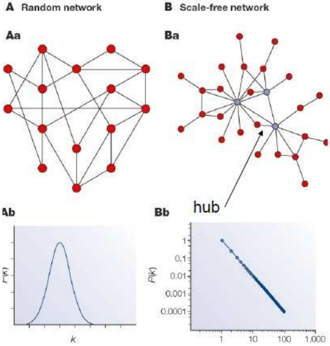
- Degree of node = number of neighbours
- Hierarchical= a lot of difference in degree?
- $P(k)$ =probability of degree k

Random network vs scale-free network



- random: $P(k)$ normally distributed

Random network vs scale-free network



- random: $P(k)$ normally distributed
- scale free: $P(k) \sim k^{-\lambda}$: power-law

Clustering coefficient

- But: having lots of neighbours $\neq$ having influence, being leader

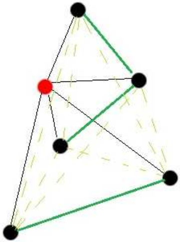
Clustering coefficient

- But: having lots of neighbours $\neq$ having influence, being leader
- Looking at clustering coefficient $C(v)$ of node v: how good neighbours are connected

Clustering coefficient

$$C(v) = \frac{|\text{edges between neighbours}|}{|\text{total possible edges between neighbours}|} = \frac{n_v}{\frac{k(k-1)}{2}}$$

with $n_v = |\text{edges between neighbours}|$; $k =$number of neighbours.



Hierarchical vs scale-free

- low clustering coefficient: neighbours depend on node, "leader"

Hierarchical vs scale-free

- low clustering coefficient: neighbours depend on node, "leader"
- high clustering coefficient: node is interchangeable with neighbours, "follower"

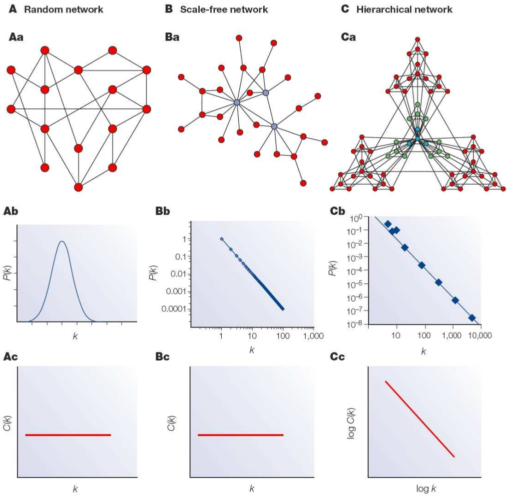
Hierarchical vs scale-free

- low clustering coefficient: neighbours depend on node, "leader"
- high clustering coefficient: node is interchangeable with neighbours, "follower"
- hierarchical network: higher degree, lower clustering coefficient

Hierarchical vs scale-free

- low clustering coefficient: neighbours depend on node, "leader"
- high clustering coefficient: node is interchangeable with neighbours, "follower"
- hierarchical network: higher degree, lower clustering coefficient
- scale-free network: clustering coefficient the same for every degree

Summary


A Random network
B Scale-free network
C Hierarchical network
Aa
Ba
Ca
Ab
Bb
Cb
P(k)
k
1
0.1
0.01
0.001
0.0001
10
100
1,000
10,000
Ac
Bc
Cc
C(k)
log C(k)
log k

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path
- Failure: nodes deleted randomly

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path
- Failure: nodes deleted randomly
- Attack: delete most connected nodes

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path
- Failure: nodes deleted randomly
- Attack: delete most connected nodes
- random: diameter increases, same for failure and attack

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path
- Failure: nodes deleted randomly
- Attack: delete most connected nodes
- random: diameter increases, same for failure and attack
- scale-free: diameter unchanged for failure; for attack it increases with bigger scope then random

Failure and attack

- Diameter of network = biggest distance between two nodes; distance= length of smallest path
- Failure: nodes deleted randomly
- Attack: delete most connected nodes
- random: diameter increases, same for failure and attack
- scale-free: diameter unchanged for failure; for attack it increases with bigger scope then random
- Civil war: attack in hierarchical network →fall apart → new highest fight for leadership. Not because hierarchy is necessary, but because hierachical structure is still present

Changing network

Find easiest way from hierarchical to scale-free network

- two networks with same number of nodes (and edges)

Changing network

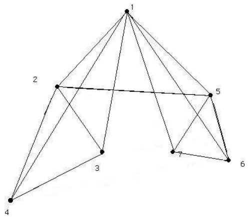
Find easiest way from hierarchical to scale-free network

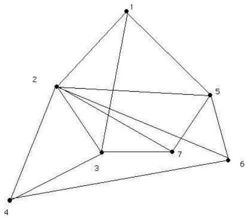
- two networks with same number of nodes (and edges)
- Find best mapping from nodes of one network to the other

Changing network

Find easiest way from hierarchical to scale-free network

- two networks with same number of nodes (and edges)
- Find best mapping from nodes of one network to the other
- best= least edges to add/remove





Changing network

Find easiest way from hierarchical to scale-free network

- Algoritm to equalize clustercoefficients:

Changing network

Find easiest way from hierarchical to scale-free network

- Algoritm to equalize clustercoefficients:
- Look at node v, do for all neighbours b:

Changing network

Find easiest way from hierarchical to scale-free network

- Algoritm to equalize clustercoefficients:
- Look at node v, do for all neighbours b:
    - $C(v) < C(b)$: destroy edge $\{v, b\}$, unless $degree(v) < k$ (k some treshold): then connect neighbour of b that isn't connected yet to v (if possible)

Changing network

Find easiest way from hierarchical to scale-free network

- Algoritm to equalize clustercoefficients:
- Look at node v, do for all neighbours b:
  - $C(v) < C(b)$: destroy edge $\{v, b\}$, unless $degree(v) < k$ (k some treshold): then connect neighbour of b that isn't connected yet to v (if possible)
  - $C(v) > C(b)$: create edge between v and neighbour of b not yet connected with v

Changing network

Find easiest way from hierarchical to scale-free network

- Algoritm to equalize clustercoefficients:
- Look at node v, do for all neighbours b:
  - $C(v) < C(b)$: destroy edge $\{v, b\}$, unless $degree(v) < k$ (k some treshold): then connect neighbour of b that isn't connected yet to v (if possible)
  - $C(v) > C(b)$: create edge between v and neighbour of b not yet connected with v
- algoritm still needs to be optimized and implemented/tested

Changing network

Build scale-free network

- Start from small graph

Changing network

Build scale-free network

- Start from small graph
- Add nodes, with m edges (m mostly between 1 and 5)

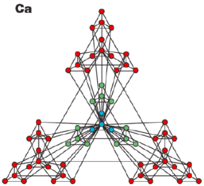
Changing network

Build scale-free network

- Start from small graph
- Add nodes, with m edges (m mostly between 1 and 5)
- Preferable attachment: more chance to connect with nodes with higher degree

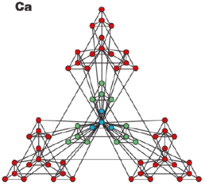
Changing network

- Build hierarchical network: Leader of cluster connected with everything in cluster and leader of other clusters, rest not connected with other clusters



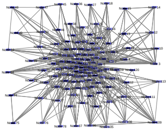
Changing network

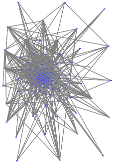
- Build hierarchical network: Leader of cluster connected with everything in cluster and leader of other clusters, rest not connected with other clusters

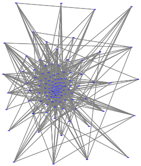


- Build random network: randomly choose two nodes and connect it, repeat for the number of edges you want

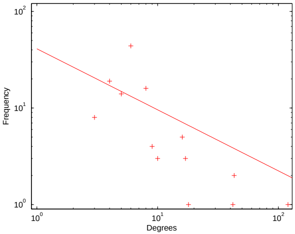
Networks

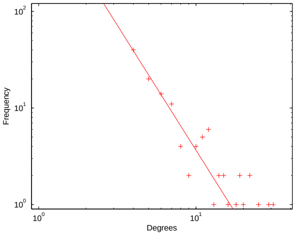


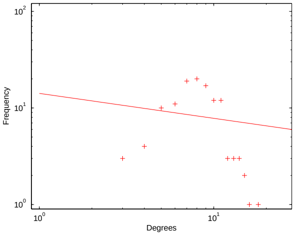




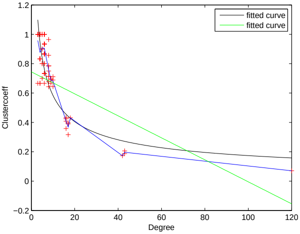
PLplot

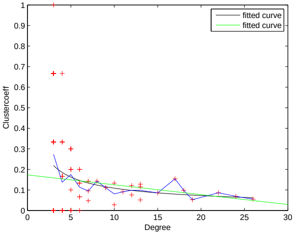

Frequency
Degrees
10^0
10^1
10^2

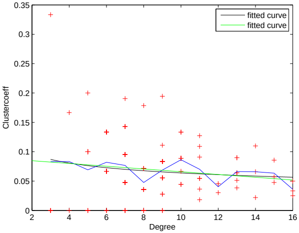

Frequency
Degrees
10^0
10^1
10^2


Frequency
Degrees
10^0
10^1
10^2

Degree-Clustercoefficient







Organization

- Organization = structure with function
  $\rightarrow$ reach a global goal, global pattern

Organization

- Organization = structure with function
  $\rightarrow$ reach a global goal, global pattern
- Self-organization: function, global activitity arise spontaneously: by local interactions, common goal put by collective

Organization

- Organization = structure with function
  $\rightarrow$ reach a global goal, global pattern
- Self-organization: function, global activitity arise spontaneously: by local interactions, common goal put by collective
- Hierarchical: structure and function (organization) decided from above, 1 agent put the common goal

Coordination

- Coordination: structuring of actions to
  - minimize friction
  - maximize synergy

Coordination

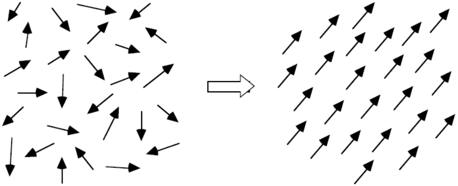
- Coordination: structuring of actions to
  - minimize friction
  - maximize synergy
- $\rightarrow$ 4 processes:
  - alignment
  - division of labor
  - workflow
  - aggregation

Alignment

- Alignment= Aim at same target to avoid friction

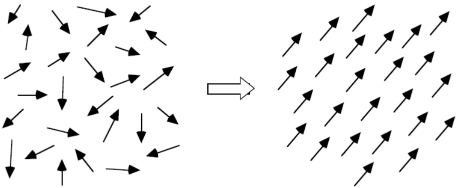
Alignment

- Alignment= Aim at same target to avoid friction
  - Self-organization: agents adapt towards neighbours by variation+selection



Alignment

- Alignment= Aim at same target to avoid friction
  - Self-organization: agents adapt towards neighbours by variation+selection



  - Hierarchical: agents adapt towards one leader

Model

- represented by graph

Model

- represented by graph
- each node i get number n_i between 0 and 1, represented by color on greyscale (0=black;1=white)

Model

- represented by graph
- each node i get number n_i between 0 and 1, represented by color on greyscale (0=black;1=white)
- color of each node moves toward neighbourcolors

$$n_i = n_i + \frac{\sum_{j \in N_i} (n_j - n_i)}{2 \cdot |N_i|}$$

Model

- represented by graph
- each node i get number n_i between 0 and 1, represented by color on greyscale (0=black;1=white)
- color of each node moves toward neighbourcolors

$$n_i = n_i + \frac{\sum_{j \in N_i} (n_j - n_i)}{2 \cdot |N_i|}$$

- higher fitness if less variation with neighbours (less friction)

$$f(n_i) = \sqrt{\frac{|N_i|}{\sum_{j \in N_i} (n_i - n_j)^2}}$$

Expectations

- We expect/hope:

Expectations

- We expect/hope:
  - hierarchical network: all colors move towards "leading" color, who gets the fittest

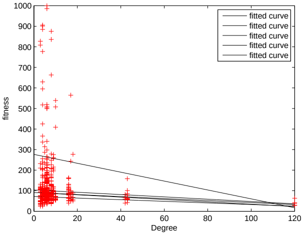
Expectations

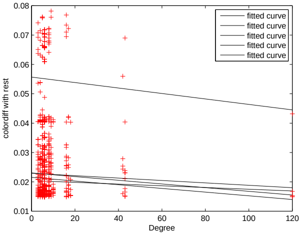
- We expect/hope:
  - hierarchical network: all colors move towards "leading" color, who gets the fittest
  - non-hierarchical network: all colors change the same amount, and all nodes have same fitness

Expectations

- We expect/hope:
  - hierarchical network: all colors move towards "leading" color, who gets the fittest
  - non-hierarchical network: all colors change the same amount, and all nodes have same fitness
- create plots with degree/clustercoeff against fitness/difference in color between node and rest

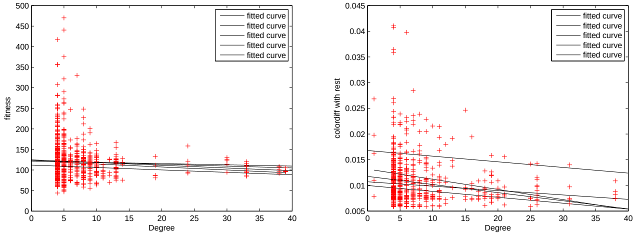
Degree-hierarchical network





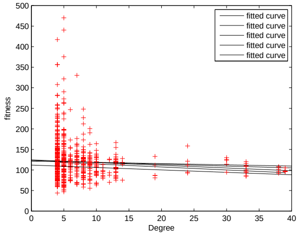
- Higher degree, lower fitness and colordifference

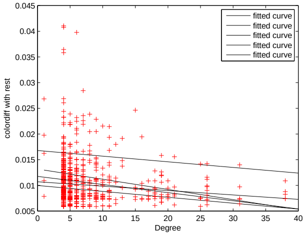
Degree-scale-free network



- Fitness independent of degree

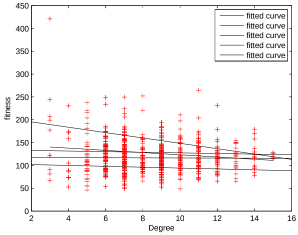
Degree-scale-free network

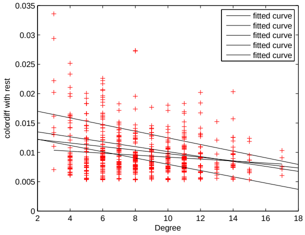




- Fitness independent of degree
- higher degree, lower colordiff

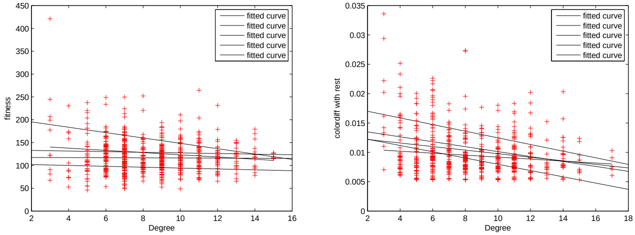
Degree-random network





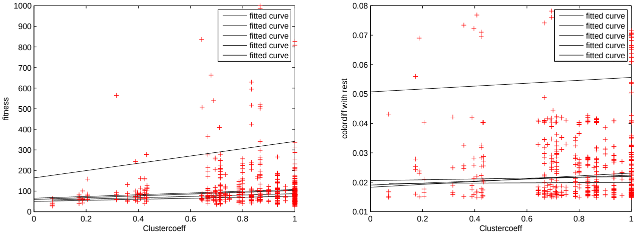
- Fitness independent of degree

Degree-random network



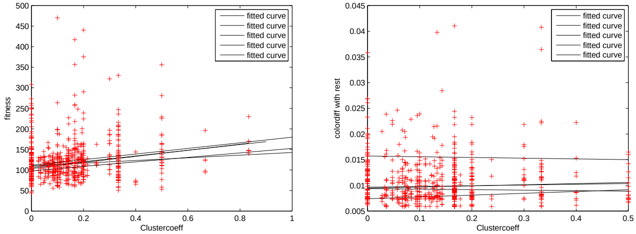
- Fitness independent of degree
- Higher degree, lower colordiff

Clustercoeff-hierarchical network



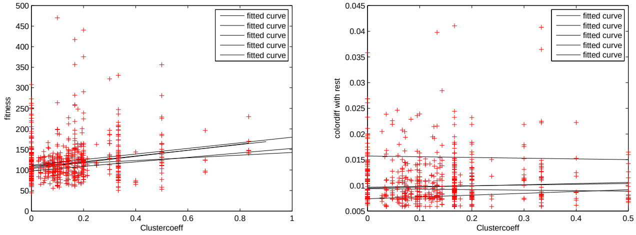
- Higher clustercoefficient, higher fitness and colordiff

Clustercoeff-scale-free network



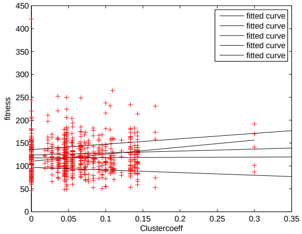
- Higher clustercoefficient, higher fitness

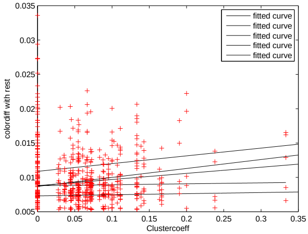
Clustercoeff-scale-free network



- Higher clustercoefficient, higher fitness
- Colordiff independent of clustercoeff

Clustercoeff-random network





- Fitness and colordiff independent of clustercoefficient

Some numbers

- mean fitness: 131.7; 124.4; 92.0 for resp hierarchical, scale-free and random network $\rightarrow$ much lower for random network

Some numbers

- mean fitness: 131.7; 124.4; 92.0 for resp hierarchical, scale-free and random network $\rightarrow$ much lower for random network
- standard deviation fitness: 60.3; 38.1; 21.4 for resp hierarchical, scale-free and random network $\rightarrow$ particulary higher for hierarchical network

Some numbers

- mean fitness: 131.7; 124.4; 92.0 for resp hierarchical, scale-free and random network → much lower for random network
- standard deviation fitness: 60.3; 38.1; 21.4 for resp hierarchical, scale-free and random network → particulary higher for hierarchical network
- mean colordiff: 0.0353; 0.0126; 0.0078 for resp hierarchical, scale-free and random network → more difference in color in hierarchical

Some numbers

- mean fitness: 131.7; 124.4; 92.0 for resp hierarchical, scale-free and random network → much lower for random network
- standard deviation fitness: 60.3; 38.1; 21.4 for resp hierarchical, scale-free and random network → particulary higher for hierarchical network
- mean colordiff: 0.0353; 0.0126; 0.0078 for resp hierarchical, scale-free and random network → more difference in color in hierarchical
- standard deviation difference in color: 0.0090; 0.0048; 0.0039 for resp hierarchical, scale-free and random network → more variation in colordiff for hierarchical network

Summary

- Hierarchical: higher degree, lower fitness. Rest: independent

Summary

- Hierarchical: higher degree, lower fitness. Rest: independent
- Higher degree, lower colordifference

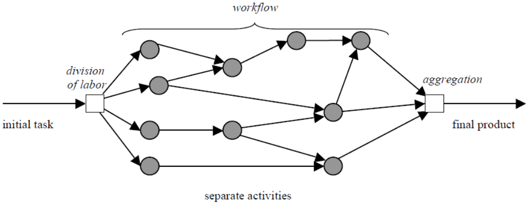
Summary

- Hierarchical: higher degree, lower fitness. Rest: independent
- Higher degree, lower colordifference
- Higher clustercoeff, higher fitness, except random

Summary

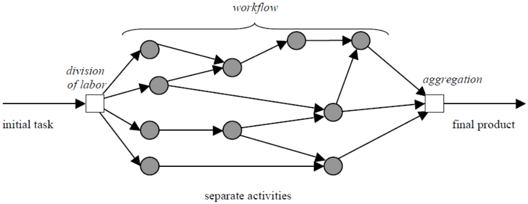
- Hierarchical: higher degree, lower fitness. Rest: independent
- Higher degree, lower colordifference
- Higher clustercoeff, higher fitness, except random
- Hierarchical: higher clustercoefficient, higher colordifference. Rest:independent

Maximize synergy



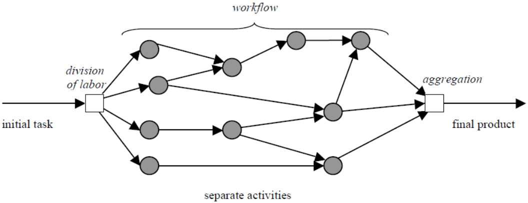
- Division of labor

Maximize synergy



- Division of labor
- Workflow

Maximize synergy



- Division of labor
- Workflow
- Aggregation

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species
- Aggregation:

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species
- Aggregation:
  - shared medium. Example: ant pheromones

Selforganization in

- Division of labor + workflow : pick task most skilled at.
  Example: evolution of different species
- Aggregation:
  - shared medium. Example: ant pheromones
  - products of different activitities interact
    example: ecosystem: other species provide resources, services

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species
- Aggregation:
  - shared medium. Example: ant pheromones
  - products of different activitities interact
    example: ecosystem: other species provide resources, services
- Again by variation-selection:

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species
- Aggregation:
    - shared medium. Example: ant pheromones
    - products of different activitities interact
      example: ecosystem: other species provide resources, services
- Again by variation-selection:
    - division of labor: variation →some more skilled → select task → get better at it

Selforganization in

- Division of labor + workflow : pick task most skilled at. Example: evolution of different species
- Aggregation:
    - shared medium. Example: ant pheromones
    - products of different activitities interact
      example: ecosystem: other species provide resources, services
- Again by variation-selection:
    - division of labor: variation →some more skilled → select task → get better at it
    - aggregation: random interaction, best are selected

Model

- inspired by ecosystem, with agents who need products and produce products others can use

Model

- inspired by ecosystem, with agents who need products and produce products others can use
- graph, each node (agent) has a vector of length n

Model

- inspired by ecosystem, with agents who need products and produce products others can use
- graph, each node (agent) has a vector of length n
- is foodvector, with m ones on it, the foodproducts the node need (rest is zero)

Model

- inspired by ecosystem, with agents who need products and produce products others can use
- graph, each node (agent) has a vector of length n
- is foodvector, with m ones on it, the foodproducts the node need (rest is zero)
- the products get "garbageproducts", by a random permutation of the foods (so each product become another product); same for all nodes

Model

- inspired by ecosystem, with agents who need products and produce products others can use
- graph, each node (agent) has a vector of length n
- is foodvector, with m ones on it, the foodproducts the node need (rest is zero)
- the products get "garbageproducts", by a random permutation of the foods (so each product become another product); same for all nodes
- with these two vectors, we can generate a garbagevector for each node

Model

- fitness of node = number of garbageproducts of neighbours that are foodproducts of node

Model

- fitness of node = number of garbageproducts of neighbours that are foodproducts of node
- simple evolution: for each node we generate 10 random fluctuations of foodvector (variation), and choose the best (selection)

Model

- fitness of node = number of garbageproducts of neighbours that are foodproducts of node
- simple evolution: for each node we generate 10 random fluctuations of foodvector (variation), and choose the best (selection)
- Colordiff → Difference in food of node with rest
  Bad variable: you can't influence food by food → don't use it

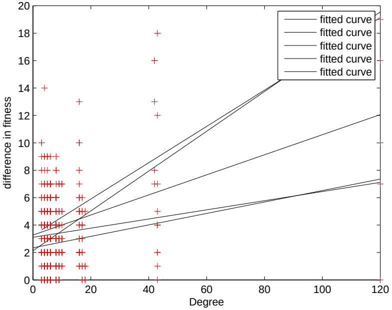
Model

- fitness of node = number of garbageproducts of neighbours that are foodproducts of node
- simple evolution: for each node we generate 10 random fluctuations of foodvector (variation), and choose the best (selection)
- Colordiff $\rightarrow$ Difference in food of node with rest
  Bad variable: you can't influence food by food $\rightarrow$ don't use it
- for each sort of network (random, scale-free, hierarchical), we plot clustercoefficient and degree against difference in fitness

Model

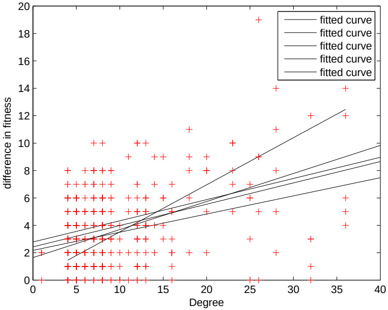
- fitness of node = number of garbageproducts of neighbours that are foodproducts of node
- simple evolution: for each node we generate 10 random fluctuations of foodvector (variation), and choose the best (selection)
- Colordiff $\rightarrow$ Difference in food of node with rest
  Bad variable: you can't influence food by food $\rightarrow$ don't use it
- for each sort of network (random, scale-free, hierarchical), we plot clustercoefficient and degree against difference in fitness
- the bigger your fitnessdifference, the more you got stronger

Degree-hierarchical network



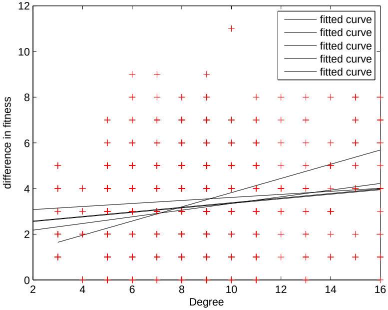
- Higher degree, higher difference in fitness

Degree-scale-free network



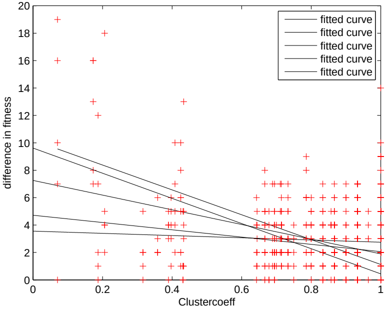
- Higher degree, higher difference in fitness, but less then hierarchical

Degree-random network



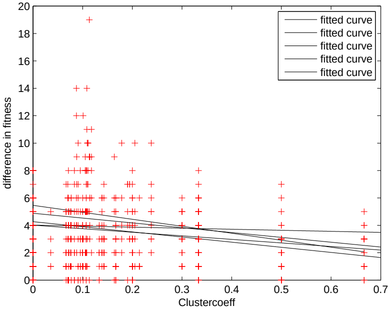
- Higher degree, a little bit higher difference in fitness

Clustercoeff-hierarchical network



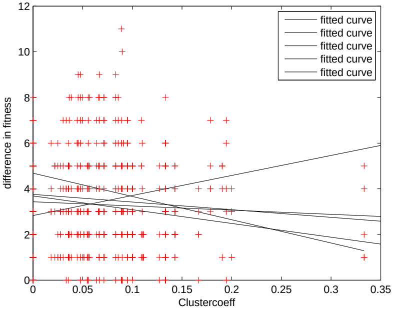
- Higher clustercoefficient, lower difference in fitness

Clustercoeff-scale-free network



- Higher clustercoefficient, a little bit lower difference in fitness

Clustercoeff-random network



- Higher clustercoefficient, a little bit lower difference in fitness

Some numbers

- mean difference in fitness: $3.21; 3.17; 3.07$ for resp hierarchical, scale-free and random network $\rightarrow$ a bit lower for random network

Some numbers

- mean difference in fitness: $3.21; 3.17; 3.07$ for resp hierarchical, scale-free and random network $\rightarrow$ a bit lower for random network
- standard deviation difference in fitness: $3.15; 2.58; 2.24$ for resp hierarchical, scale-free and random network $\rightarrow$ a bit higher for hierarchical network

Summary

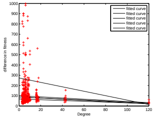
- Higher degree, higher difference in fitness; more if hierarchical network, and only a little bit for random network

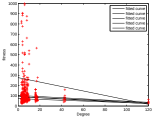
Summary

- Higher degree, higher difference in fitness; more if hierarchical network, and only a little bit for random network
- Higher clustercoefficient, lower difference in fitness; only a little bit for scale-free and random

Why not look at...

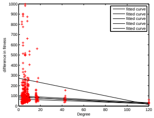
- Difference in fitness (fitness in beginning different?)? Equivalent to fitness:

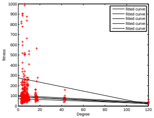




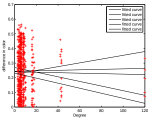
Why not look at...

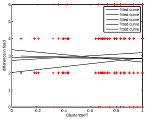
- Difference in fitness (fitness in beginning different?)? Equivalent to fitness:





- Difference in color/food between end and beginning? Independent of degree/clustercoeff:





Fitness

- The overal fitness of the network is:
  a little bit higher for hierarchical network, particulary lower for random network.

Fitness

- The overal fitness of the network is:
  a little bit higher for hierarchical network, particulary lower for random network.
- The unequality of fitness of the network is:
  higher for hierarchical network, a bit lower for random network

Fitness

- The overal fitness of the network is:
  a little bit higher for hierarchical network, particulary lower for random network.
- The unequality of fitness of the network is:
  higher for hierarchical network, a bit lower for random network
- So intuition that hierachical network= more unequality, holds.

Influence

- In the first model, the overal difference in color with the rest of the network is:
  a little bit higher for hierarchical network, particulary lower for random network

Influence

- In the first model, the overal difference in color with the rest of the network is:
  a little bit higher for hierarchical network, particulary lower for random network
- Unequality of infuence: higher in hierarchical network

Influence

- In the first model, the overal difference in color with the rest of the network is:
  a little bit higher for hierarchical network, particulary lower for random network
- Unequality of infuence: higher in hierarchical network
- So while nodes in a hierarchical network agree more with their neighbours, they do less with the whole network

Strange things

- Higher degree/lower clustercoeff $\rightarrow$ lower fitness in hierarchical network of 1st model: clusters evolve independent, leading node is 'in between' (only local problem)

Strange things

- Higher degree/lower clustercoeff $\rightarrow$ lower fitness in hierarchical network of 1st model: clusters evolve independent, leading node is 'in between' (only local problem)
- Hierarchical properties of scale-free network: the network used isn't completely scale-free

Normal things

- Higher degree $\rightarrow$ lower colordifference in first model: Leaders differ less with rest

Normal things

- Higher degree $\rightarrow$ lower colordifference in first model: Leaders differ less with rest
- Higher degree/Lower clustercoeff $\rightarrow$ higher fitness in second model: Leaders become fitter, particulary in hierarchical network

Further research

- Work with directed graph;
  - Hierarchy= graph represents order
  - Look at what it does in the models

Further research

On second model:

- Use something else then fooddiff (garbage in it)

Further research

On second model:

- Use something else then fooddiff (garbage in it)
- Complex evolution: choose partner by fitness, in the group of neighbours with enough in common, cross-over and mutate (based on genetic algorithm)

Further research

On second model:

- Use something else then fooddiff (garbage in it)
- Complex evolution: choose partner by fitness, in the group of neighbours with enough in common, cross-over and mutate (based on genetic algorithm)
- Avoide friction: fitness lower if garbage is used by more nodes, higher if more neighbours with garbage

Further research

- Changing network:
  - Get children by GA, connected to parents and same fitness as parent (mothermilk/feeded by parent)

Further research

- Changing network:
  - Get children by GA, connected to parents and same fitness as parent (mothermilk/feeded by parent)
  - Connect to neighbours of neighbours, more chance if more increase of fitness

Further research

- Changing network:
  - Get children by GA, connected to parents and same fitness as parent (mothermilk/feeded by parent)
  - Connect to neighbours of neighbours, more chance if more increase of fitness
  - Dying:
    - Some garbage is "vital", neighbour can eat you, depending on your and his fitness
    - The lower fitness, the more chance to die

The end

Thank you!