

Artificial Life Summer 2025

Self Organized Criticality, SOC Ant Algorithms

Master Computer Science [MA-INF 4201]

Mon 14:15 – 15:45, HSZ, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Examination:

After the end of the lecture period (18.7.25) there will be an examination for the module Artificial Life.

To be admitted for the exam , you will need

- to register
- a minimum of **50%** of the possible reachable points
- a minimum **two presentations** of your solutions in the exercises.

Exam date: Tuesday **22 July 2025, 12:00**, Lecture Hall 1+2, HSZ

The examination will be a written exam,
operated in presence, Lecture Hall 1+2, HSZ
exam duration is 100 minutes
up to 100 points are reachable (approx. 1 minute per point).

Some details for the exam

Exam:

Tuesday 22.7.25 12:00 - 15:00

Lecture Hall 1 and Lecture Hall 2

Date for resit exam:

Monday 8.9.25 9:00 - 12:00

Lecture Hall 1

Topics

- **Self Organized Criticality, SOC**
- Ant Algorithms

Overview: SOC

- What is Self Organized Criticality?
- Motivation
- Power law , Scaling law
- Examples of SOC systems
 - Forest Fire Model
 - Sandpile model
 - Land slides
 - Earthquakes
 - Zipf's Law
 - Lotka's Law
 - Size Distributions of Cities

Motivation:

Within the scientific literature **Self Organizing Criticality** is sometimes denoted as:

Self Organization to Criticality
Self-Organized Criticality

Motivation:

Within the scientific literature **Self Organizing Criticality** is sometimes denoted as:

Self Organization to Criticality
Self-Organized Criticality

Examples for **SOC** systems have been reported from:

- Statistical Physics
- Nonlinear dynamical systems,
- Theory of Self Organization
- Economics
- Systems biology
- Bio-Inspired computational intelligence
- Artificial Life

Motivation:

Self Organizing Criticality (SOC) is a special property that can be observed in some dynamical systems.

These systems show the interesting (and somehow strange) behavior, to be most of the time in a **critical** situation or state.

Motivation:

Self Organizing Criticality (SOC) is a special property that can be observed in some dynamical systems.

These systems show the interesting (and somehow strange) behavior, to be most of the time in a **critical** situation or state.

A situation is called **critical**, if a large or substantial change within the system is about to occur.

If an **event** happens, the event is changing the situation of the system in a non-reversible, drastic way.

Sometimes the change can be called *dramatic*.

Motivation:

Self Organizing Criticality (SOC) is a special property that can be observed in some dynamical systems.

These systems show the interesting (and somehow strange) behavior, to be most of the time in a **critical** situation or state.

A situation is called **critical**, if a large or substantial change within the system is about to occur.

If an **event** happens, the event is changing the situation of the system in a non-reversible, drastic way.

Sometimes the change can be called *dramatic*.

In other words, since these systems are in a critical state, these systems are typically close to a catastrophe.

Motivation:

The dynamics of systems that can show Self Organizing Criticality (SOC) generates **events**.

Motivation:

The dynamics of systems that can show **Self Organizing Criticality (SOC)** generates **events**.

An **event** is a discrete, finite change of the system state, with a well defined starting and a well defined ending.

Motivation:

The dynamics of systems that can show **Self Organizing Criticality (SOC)** generates **events**.

An **event** is a discrete, finite change of the system state, with a well defined starting and a well defined ending.

Common examples of **events** are:
earthquakes, landslides, traffic jams, groups of cars in traffic flow, occurrences of keywords in text, ...

Motivation:

The dynamics of systems that can show **Self Organizing Criticality (SOC)** generates **events**.

An **event** is a discrete, finite change of the system state, with a well defined starting and a well defined ending.

Common examples of **events** are:
earthquakes, landslides, traffic jams, groups of cars in traffic flow, occurrences of keywords in text, ...

Events occur from time to time, and occur with different strength. The strength or size of an event is denoted by **s**, and the number of events is denoted by **N**.

Motivation:

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Motivation:

The dynamics of systems that can show **Self Organizing Criticality (SOC)** generates **events**.

An **event** is a discrete, finite change of the system state, with a well defined starting and a well defined ending.

Common examples of **events** are:
earthquakes, landslides, traffic jams, groups of cars in traffic flow, occurrences of keywords in text, ...

Events occur from time to time, and occur with different strength. The strength or size of an event is denoted by **s** , and the number of events is denoted by **N** .

SOC systems show an interesting relation between **s** and **N** .

Motivation:

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

The following question might give you an idea:

Motivation:

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

The following question might give you an idea:

How large is a typical earthquake?

Motivation:

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

The following question might give you an idea:

How large is a typical earthquake?

There are a lot of small earthquakes,
some with medium strength,
a few large,
and rare extreme ones.

Scaling Law

A function $f(x)$ is denoted to obey a **scaling law**, or is denoted **scaling invariant** if it shows the following property when scaling the argument x from $x \rightarrow \lambda x$:

$$f(\lambda x) \sim f(x)$$

Scaling Law

A function $f(x)$ is denoted to obey a **scaling law**, or is denoted **scaling invariant** if it shows the following property when scaling the argument x from $x \rightarrow \lambda x$:

$$\begin{aligned} f(\lambda x) &\sim f(x) \\ f(\lambda x) &= C(\lambda)f(x) \end{aligned}$$

Where the scaling factor $C(\lambda)$ does not depend on x .

Scaling Law

A function $f(x)$ is denoted to obey a **scaling law**, or is denoted **scaling invariant** if it shows the following property when scaling the argument x from $x \rightarrow \lambda x$:

$$\begin{aligned} f(\lambda x) &\sim f(x) \\ f(\lambda x) &= C(\lambda) f(x) \end{aligned}$$

Where the scaling factor $C(\lambda)$ does not depend on x .

In principle $C(\lambda)$ can be any function, but is often just an integer n .

$$f(\lambda x) = \lambda^n f(x)$$

Power Laws & Scaling Laws

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Power Laws & Scaling Laws

The relation between the strength s of an event and the frequency, or number of events N of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Size distribution:

How does the number of events $N(s)$ depend on the size s of the event?

Power Laws & Scaling Laws

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Size distribution:

How does the number of events **N(s)** depend on the size **s** of the event?

$$N(s) \sim 1 / s^{\beta}$$

Power Laws & Scaling Laws

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Size distribution:

How does the number of events **N(s)** depend on the size **s** of the event?

$$N(s) \sim 1 / s^{\beta}$$

$$\log(N(s)) \sim -\beta \log(s)$$

Power Laws & Scaling Laws

The relation between the strength **s** of an event and the frequency, or number of events **N** of SOC systems show a behavior that obeys a **power law** or a **scaling law**.

Size distribution:

How does the number of events **N(s)** depend on the size **s** of the event?

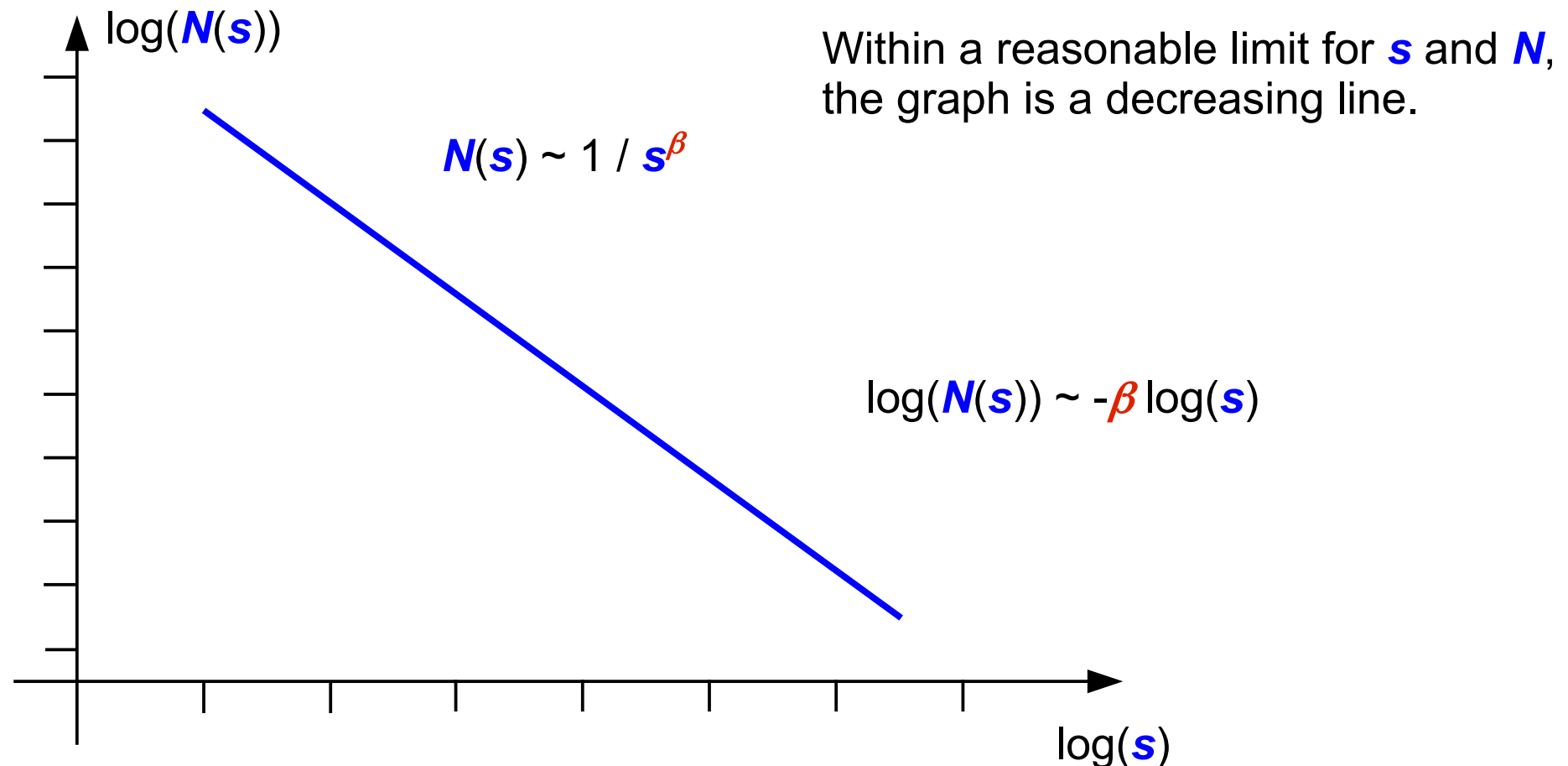
$$N(s) \sim 1 / s^{\beta}$$

$$\log(N(s)) \sim -\beta \log(s)$$

This is a linear decreasing dependency of the number of events with the strength; both in logarithmic scale.

Power Laws & Scaling Laws

The **size distribution graph** shows the characteristic, linear decreasing curve, setting the number of events $N(s)$ in relation to the size s of the event in a logarithmic plot.



Power Laws & Scaling Laws

Other aspects that are known to show a **power law** or a **scaling law** behavior:

Power Laws & Scaling Laws

Other aspects that are known to show a **power law** or a **scaling law** behavior:

Temporal distribution:

Number of events **N** depend on the time **τ** between events,
The Inter-event-interval distribution

$$N(\tau) \sim 1 / \tau^\gamma$$

Power Laws & Scaling Laws

Other aspects that are known to show a **power law** or a **scaling law** behavior:

Temporal distribution:

Number of events **N** depend on the time **τ** between events,
The Inter-event-interval distribution

$$N(\tau) \sim 1 / \tau^\alpha$$

Frequency distribution, density distribution

Power spectrum **$P(f)$** depends on the frequency **f** of the signal.

$$P(f) \sim 1 / f^\alpha$$

Examples of SOC Systems

- **Forest fire model** (K. Chen, P. Bak, M. H. Jensen: 1990
B. Drossel, F. Schwabl: 1992)
- **Sandpile model** (P. Bak, C. Tang, K. Wiesenfeld: 1987)
- **Land slides** (Y. Fuii: 1969)
- **Percolation Theory** (S. Broadbent, J. Hammersley, 1957)
- **Earthquakes** (B. Gutenberg, C.F. Richter, 1949, 1954)
- **Zipf's Law** (G.K. Zipf: 1935)
- **Lotka's Law** (A.J. Lotka: 1926)
- **Auerbach's Detection** (F. Auerbach, 1913)

SOC example: Forest-Fire Model

Forest-Fire CA

$d=2$, rectangular grid, $r=1$, von Neumann, $k=3$
non deterministic cellular automaton,

A empty / **A**shes

T Tree

F burning tree / **F**ire

 von Neumann

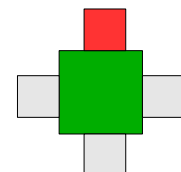
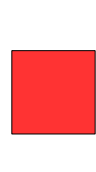
Chen, K., Bak, P. and Jensen, M. H. (1990),
"A deterministic critical forest-fire model."
 Phys. Lett. A 149, 207–210.

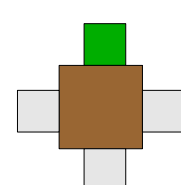
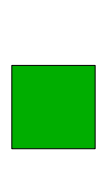
Drossel, B. and Schwabl, F. (1992)
"Self-organized critical forest-fire model."
 Phys. Rev. Lett. 69, 1629–1632.

 \rightarrow  Fire turns into ashes

 \xrightarrow{p}  Spontaneous growth

 \xrightarrow{f}  Spontaneous fire

 \rightarrow  Induced fire: a tree burns if at least one neighbor burns

 \xrightarrow{q}  Induced growth: a tree grows if at least one neighbor is a tree

SOC example: Forest-Fire Model

Forest-Fire CA

$d=2$, rectangular grid, $r=1$, von Neumann, $k=3$
non deterministic cellular automaton,

The behavior of the complete system is determined by the underlying micro-behavior of the cells.

The control parameters are:

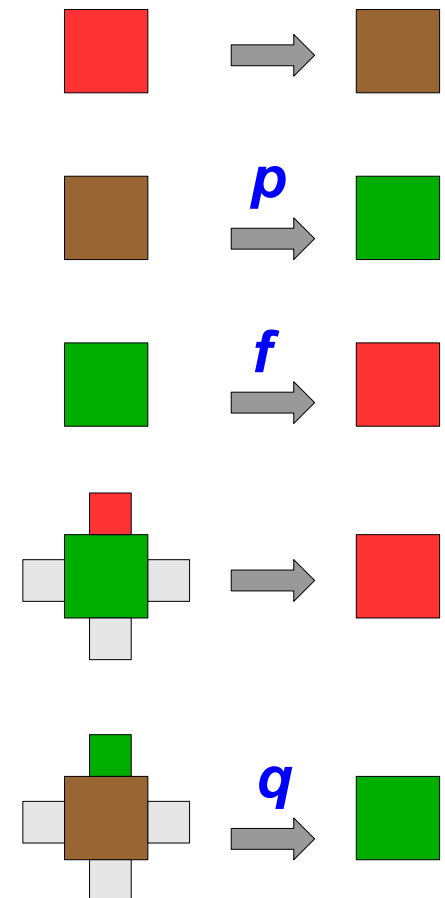
p rate of spontaneous growth

f rate of spontaneous fire

q rate of induced growth

A setting of $q=0$ (no induced growth) is a good setting to start from.

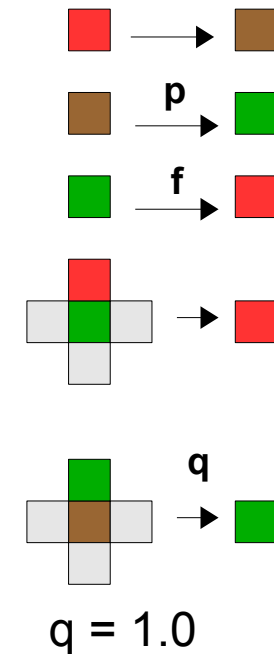
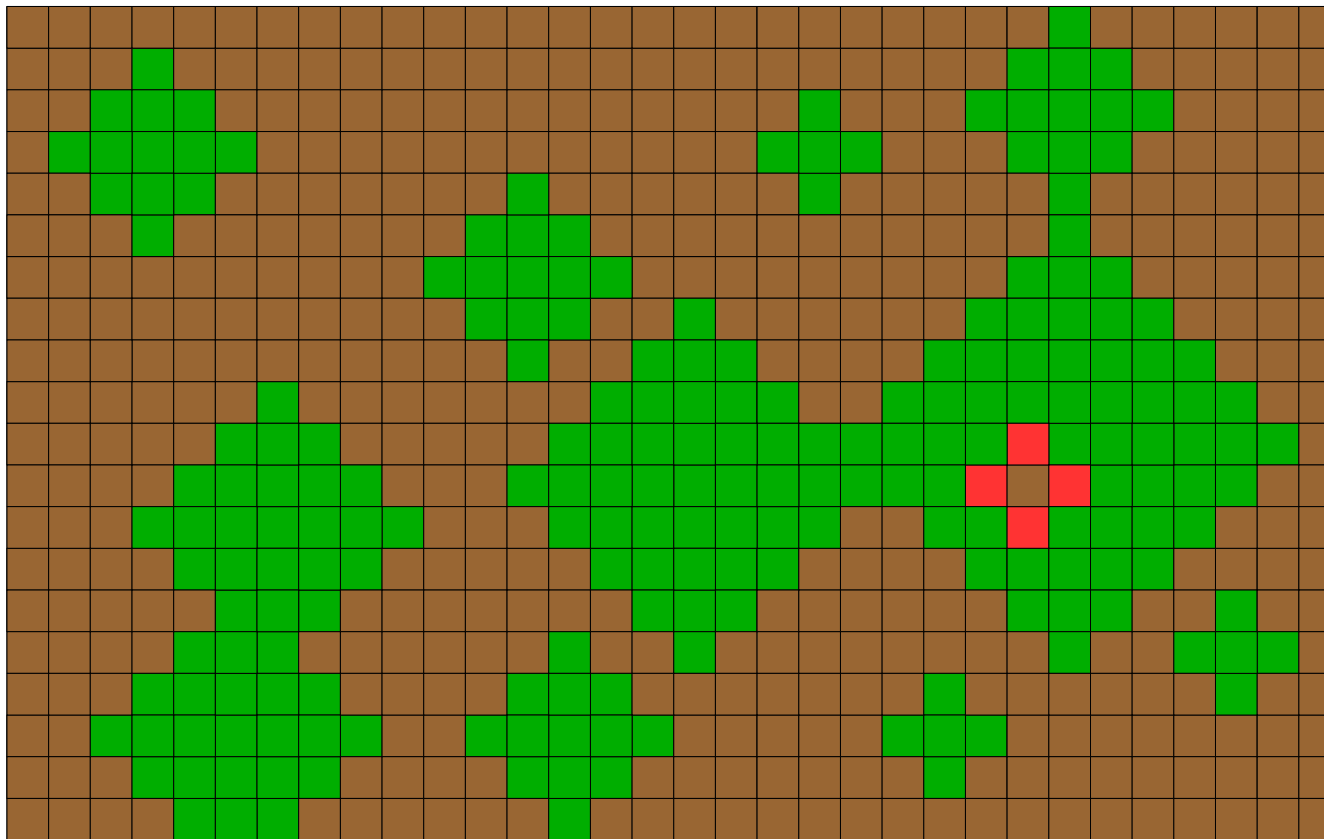
Interesting (fractal) behavior arises when $f \ll p$
 (e.g. $p/f = 100$).



SOC example: Forest-Fire Model

Forest-Fire CA

$d=2$, rectangular grid, $r=1$, von Neumann, $k=3$
non deterministic cellular automaton,



Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

A simple model that shows the effects of self organizing criticality is the **Sandpile Model**, proposed by Per Bak, Chao Tang and Kurt Wiesenfeld in 1987. They showed that such dynamical systems evolve naturally into a critical state through a self-organizational process.

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

A simple model that shows the effects of self organizing criticality is the **Sandpile Model**, proposed by Per Bak, Chao Tang and Kurt Wiesenfeld in 1987. They showed that such dynamical systems evolve naturally into a critical state through a self-organizational process.

The system consists of a regular lattice with sites that can contain “grains of sand”. The amount of grains is considered as the height $h(x)$ in position x .

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

A simple model that shows the effects of self organizing criticality is the **Sandpile Model**, proposed by Per Bak, Chao Tang and Kurt Wiesenfeld in 1987. They showed that such dynamical systems evolve naturally into a critical state through a self-organizational process.

The system consists of a regular lattice with sites that can contain “grains of sand”. The amount of grains is considered as the height $h(x)$ in position x .

First, a particle is added to a random site: $h(x) += 1$.

Second, after the addition, the difference between the height of adjacent sites determine if the grain of sand stays there, or if the configuration is instable $h(x) - h(x+1) \geq C$ and the grain tumbles aside.

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

A simple model that shows the effects of self organizing criticality is the **Sandpile Model**, proposed by Per Bak, Chao Tang and Kurt Wiesenfeld in 1987. They showed that such dynamical systems evolve naturally into a critical state through a self-organizational process.

The system consists of a regular lattice with sites that can contain “grains of sand”. The amount of grains is considered as the height $h(x)$ in position x .

First, a particle is added to a random site: $h(x) += 1$.

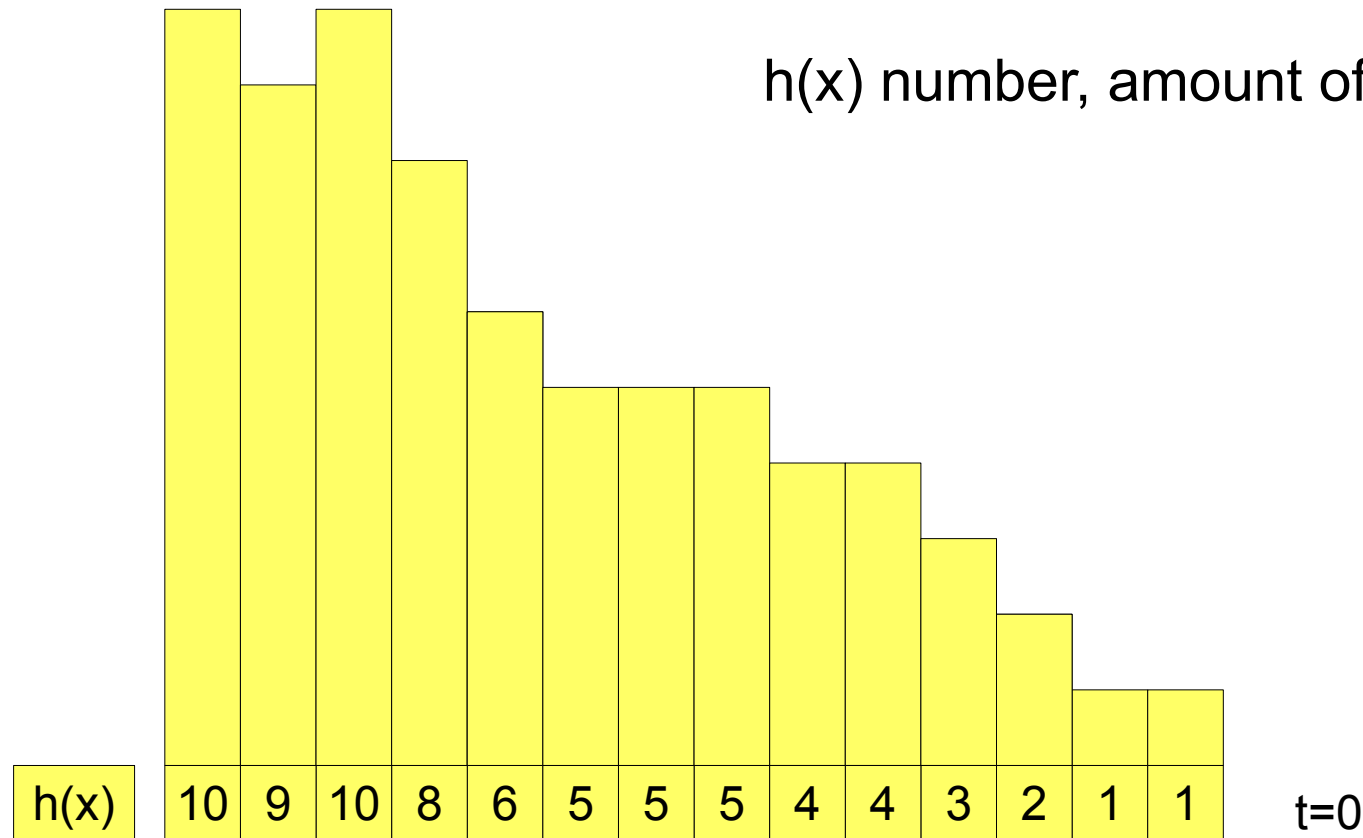
Second, after the addition, the difference between the height of adjacent sites determine if the grain of sand stays there, or if the configuration is instable $h(x) - h(x+1) \geq C$ and the grain tumbles aside.

In case the grain of sand is moving it can cause the other site to become instable as well, making other grains to move, and so on (avalanche).

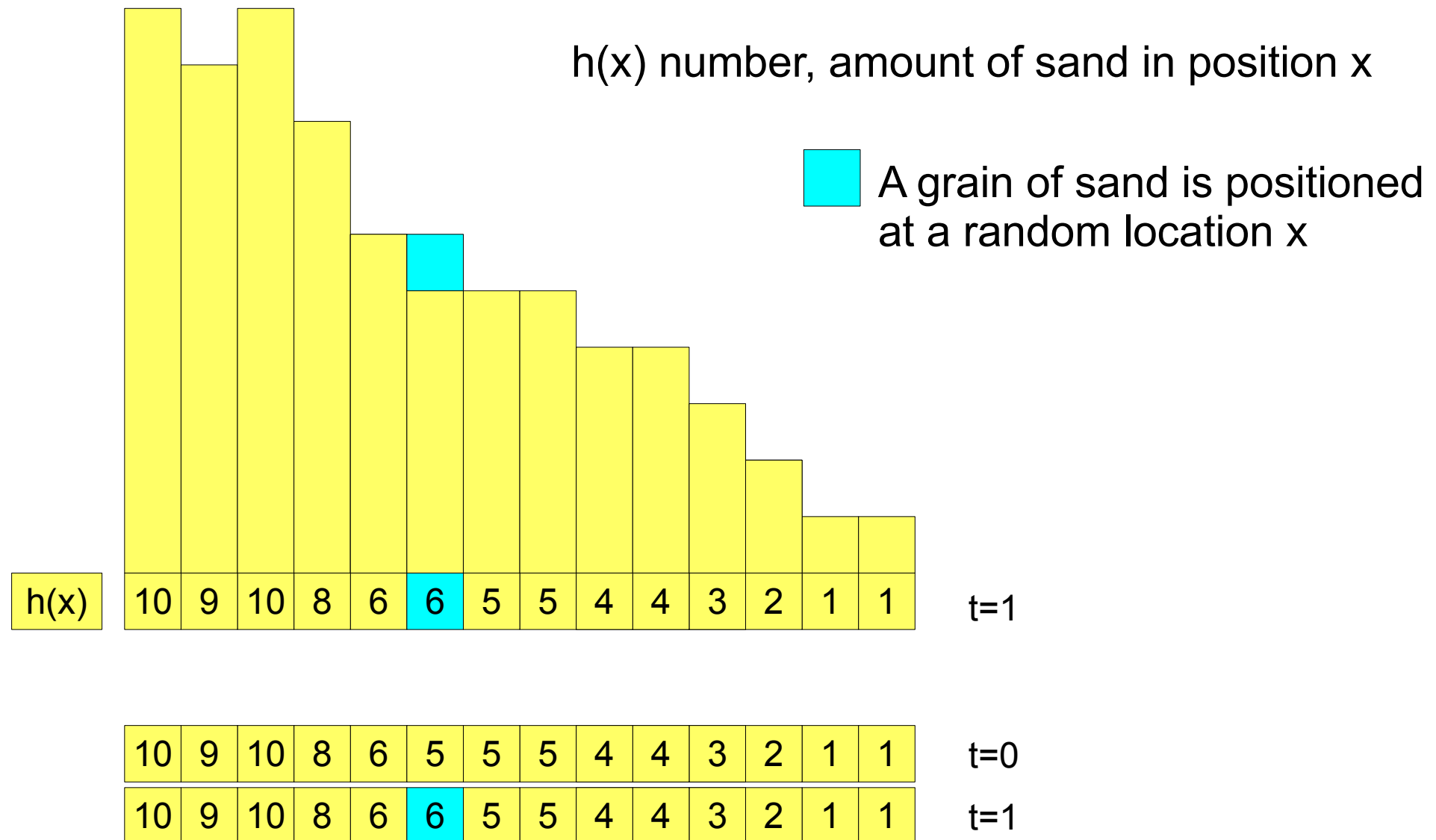
After a while the system gains a stable configuration and the next grain of sand can be positioned randomly.

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

$h(x)$ number, amount of sand in position x



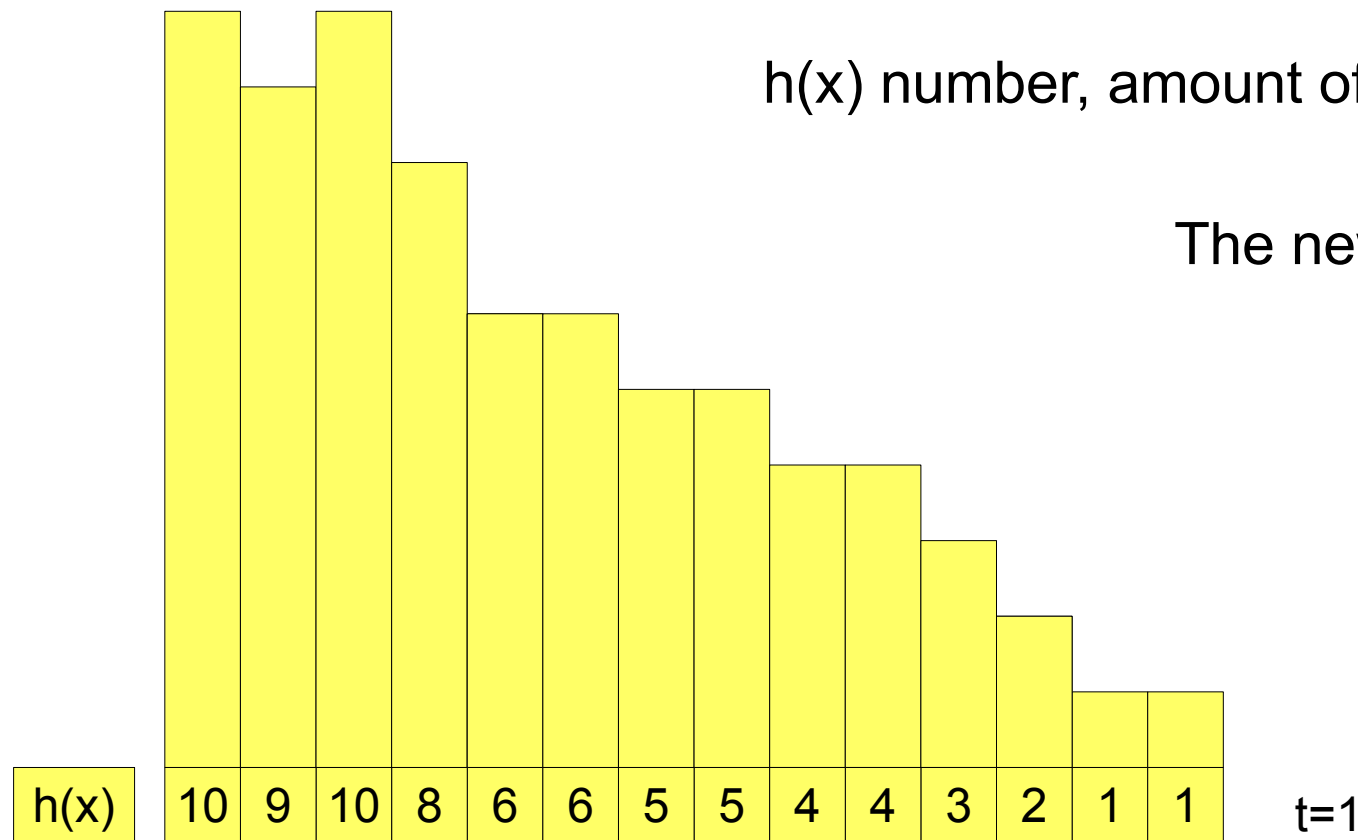
Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)



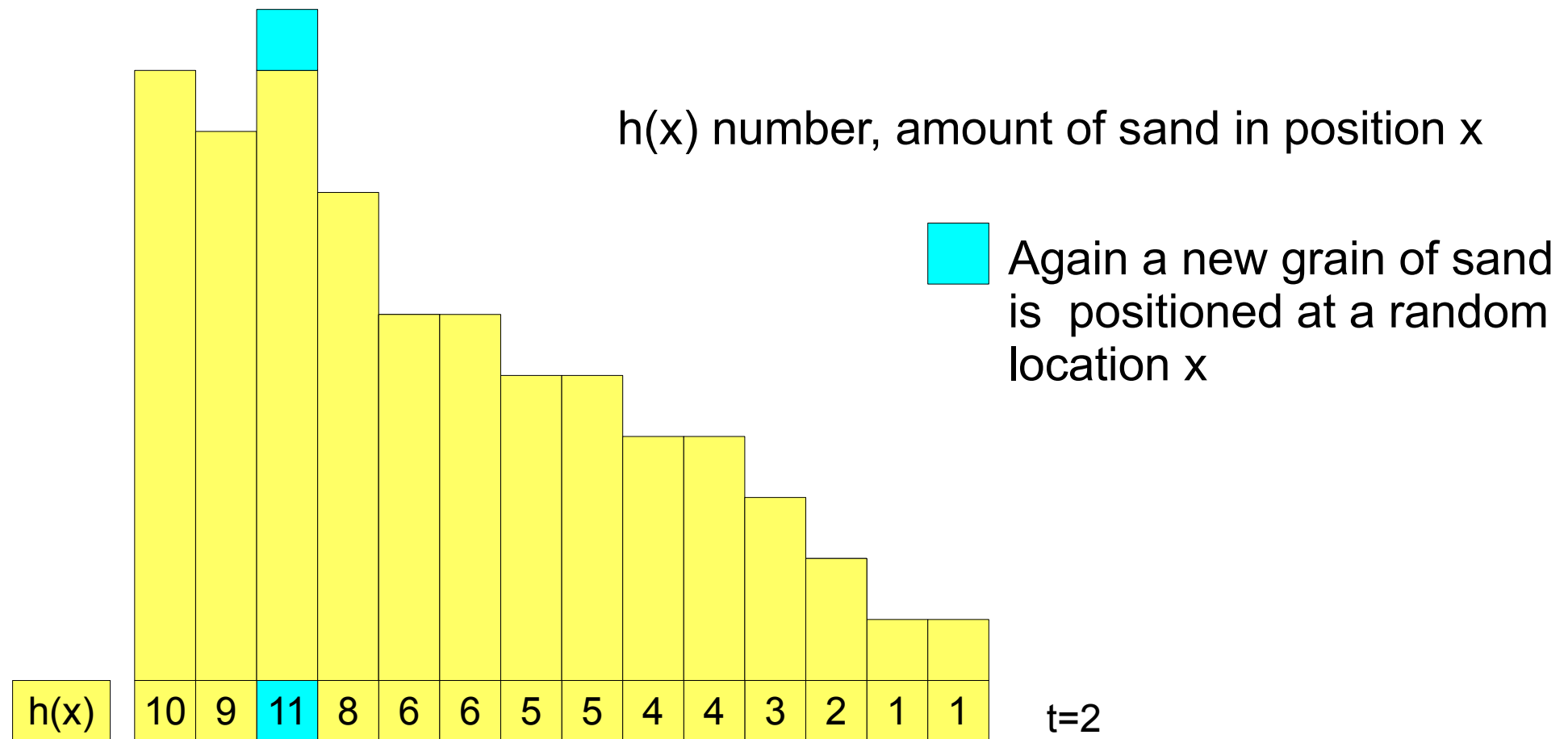
Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

$h(x)$ number, amount of sand in position x

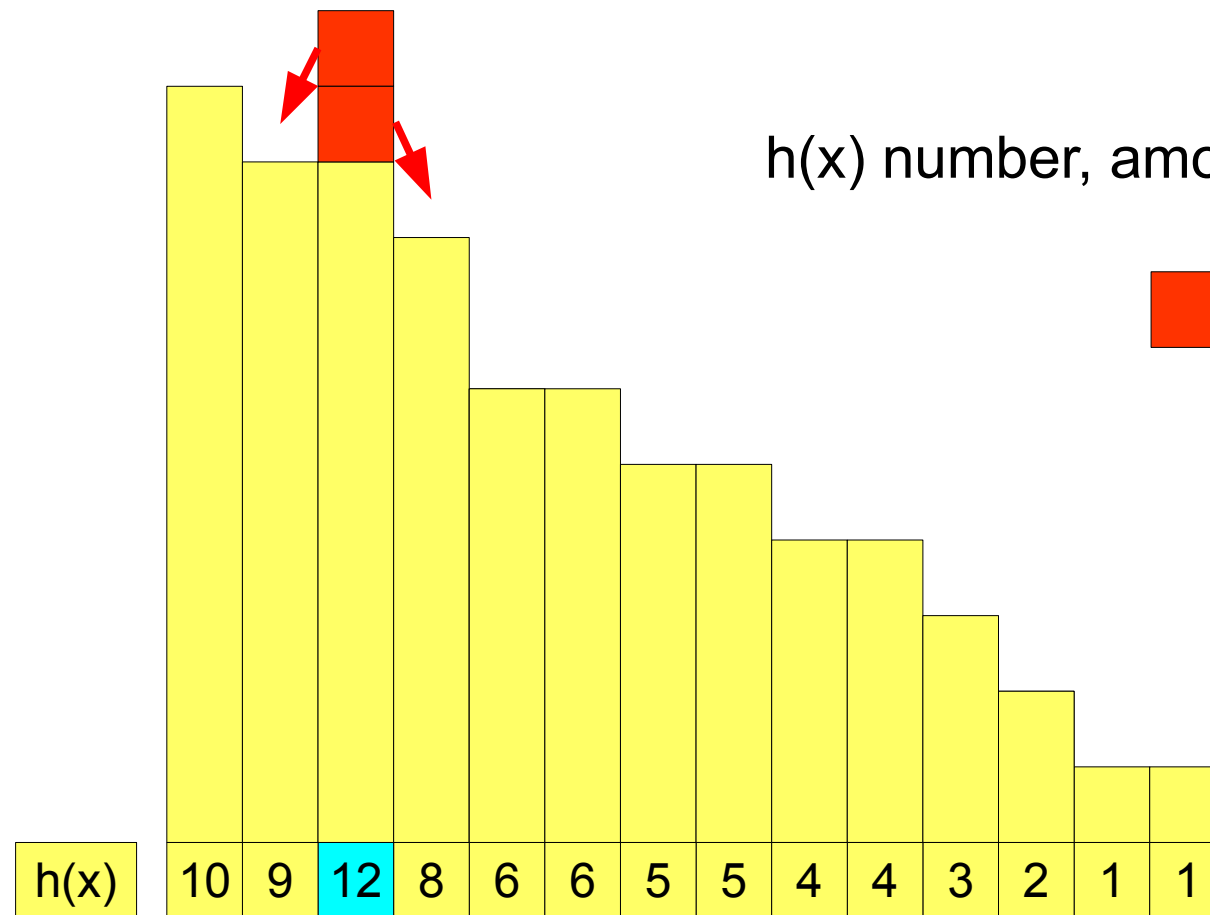
The new situation is stable.



Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)



Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)



$h(x)$ number, amount of sand in position x



This time the pile is instable,
since $h(x) > h(x+1) + 2$

The grains tumble aside:

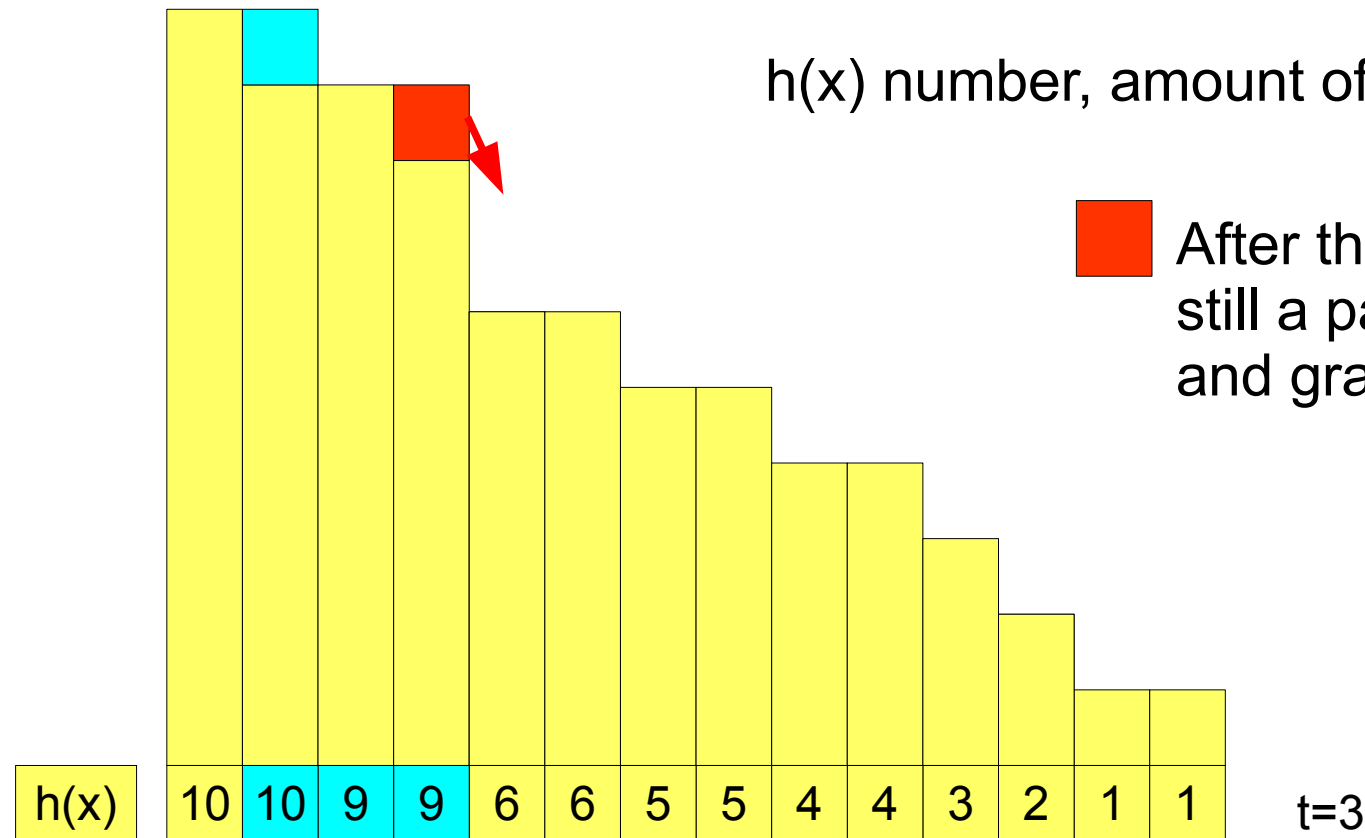
$h(x) \quad -= 2$

$h(x-1) \quad += 1$

$h(x+1) \quad += 1$

$t=2$

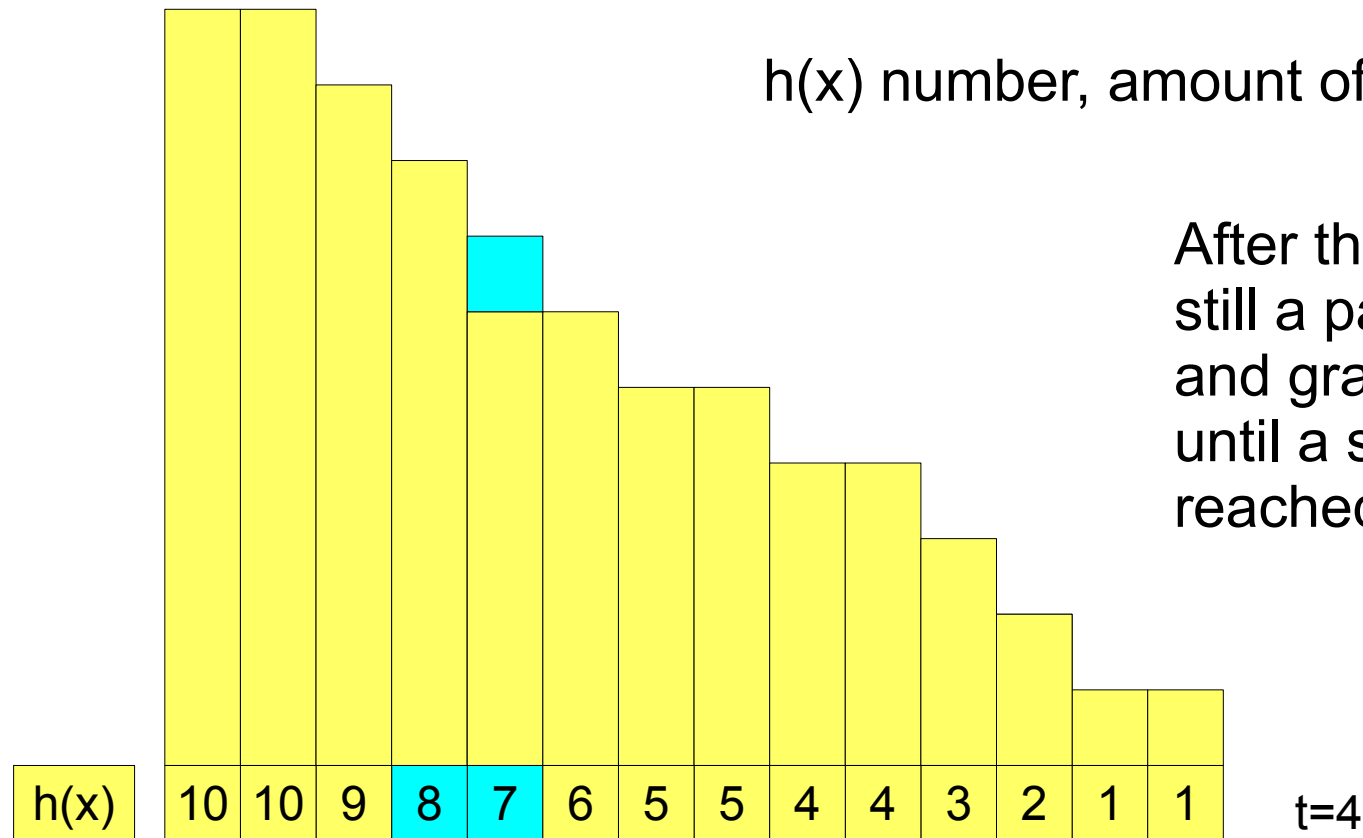
Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)



Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

$h(x)$ number, amount of sand in position x

After the movement,
still a part of the pile is instable,
and grains will continue moving,
until a stable situation is
reached again.



Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

Sandpile model in one dimension

A unit of sand is added at a random position x :

$$h(x) \rightarrow h(x) + 1$$

$$h(j) \rightarrow h(j) \quad \text{for all positions } j \neq x$$

When the height difference $h(x) - h(x+1)$ becomes higher than a fixed critical value $C=3$, the sand moves to the adjacent fields:

$$h(x) \rightarrow h(x) - 2$$

$$h(x+1) \rightarrow h(x+1) + 1$$

$$h(x-1) \rightarrow h(x-1) + 1$$

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

Instead of looking at the height $h(x)$ of the sandpile, it is easier to look at the differences between adjacent fields: $z(x) = h(x) - h(x+1)$

A unit of sand is added at a random position x :

$$h(x) \rightarrow h(x) + 1$$

$$z(x) \rightarrow z(x) + 1$$

$$z(x-1) \rightarrow z(x-1) - 1$$

When the height difference $z(x) = h(x) - h(x+1)$ becomes higher than a fixed critical value $z(x) \geq C = 3$, the sand moves to both adjacent fields:

$$z(x) \rightarrow z(x) - 3$$

$$z(x-1) \rightarrow z(x-1) + 3$$

$$z(x+1) \rightarrow z(x+1) + 1$$

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

The BTW sandpile model of Bak, Tang, Wiesenfeld was defined as a discrete, cellular based system with a rectangular grid and discrete states (almost like a cellular automaton).

Each position (cell) is representing by the value $z(x,y)$ the no of grains, or the slope within a sandpile in position (x,y) .

The critical value is set to $C=4$, which results for the stable state, that only values for the cells to be $z(x,y) \in \{0,1,2,3\}$ are possible.

Depending on the intermediate values $z(x,y)$ part of the sandpile gets instable, collapses, and grains of sand are distributed to adjacent cells ($r=1$, von Neumann neighborhood).

The distributed grains might cause further positions to get instable, they collapse, and again grains are distributed to other cells.

At the boundary of the field, grains leave the playground; “they fall off”.

Sandpile Model (P. Bak, C.Tang, K.Wiesenfeld: 1987)

- 0:** Initialize all positions, e.g. by random, or with a special pattern
- 1:** Choose a position (x,y) randomly
- 2:** Add a grain to the position (x,y), keep the rest unchanged
$$z(x,y) \rightarrow z(x,y) + 1$$
$$z(a,b) \rightarrow z(a,b) \text{ if } (a,b) \neq (x,y)$$
- 3:** Check if all positions are stable: IF ALL $z(x,y) < 4$ GOTO step 1
- 4:** Take one of the instable positions, and redistribute those grains:
$$z(x,y) \rightarrow z(x,y) - 4$$
$$z(x-1,y) \rightarrow z(x-1,y) + 1$$
$$z(x+1,y) \rightarrow z(x+1,y) + 1$$
$$z(x,y-1) \rightarrow z(x,y-1) + 1$$
$$z(x,y+1) \rightarrow z(x,y+1) + 1$$

Grains at the edge of the grid, fall off the playground, and are lost.

- 5:** GOTO Step 3 (thus repeating Step 4 until all positions are stable)

In part taken from "http://en.wikipedia.org/wiki/Abelian_sandpile_model", 5/2013

$t=0$

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=2

1	2	0	2	3
2	3	2	3	0
1	2	4	3	2
3	1	3	2	1
0	2	2	1	2

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=2

1	2	0	2	3
2	3	2	3	0
1	2	4	3	2
3	1	3	2	1
0	2	2	1	2

t=3

1	2	0	2	3
2	3	3	3	0
1	3	0	4	2
3	1	4	2	1
0	2	2	1	2

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=2

1	2	0	2	3
2	3	2	3	0
1	2	4	3	2
3	1	3	2	1
0	2	2	1	2

t=3

1	2	0	2	3
2	3	3	3	0
1	3	0	4	2
3	1	4	2	1
0	2	2	1	2

t=4

1	2	0	2	3
2	3	3	4	0
1	3	2	0	3
3	2	0	4	1
0	2	3	1	2

Adapted from:
P. Bak 96,
How nature Works,
The science of self-
organized criticality

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=2

1	2	0	2	3
2	3	2	3	0
1	2	4	3	2
3	1	3	2	1
0	2	2	1	2

t=3

1	2	0	2	3
2	3	3	3	0
1	3	0	4	2
3	1	4	2	1
0	2	2	1	2

t=4

1	2	0	2	3
2	3	3	4	0
1	3	2	0	3
3	2	0	4	1
0	2	3	1	2

t=5

1	2	0	3	3
2	3	4	0	1
1	3	2	2	3
3	2	1	0	2
0	2	3	2	2

t=6

1	2	1	3	3
2	4	0	1	1
1	3	3	2	3
3	2	1	0	2
0	2	3	2	2

t=7

1	3	1	3	3
3	0	1	1	1
1	4	3	2	3
3	2	1	0	2
0	2	3	2	2

t=8

1	3	1	3	3
3	1	1	1	1
2	0	4	2	3
3	3	1	0	2
0	2	3	2	2

t=9

1	3	1	3	3
3	1	2	1	1
2	1	0	3	3
3	3	2	0	2
0	2	3	2	2

Adapted from:
P. Bak 96,
How nature Works,
The science of self-
organized criticality

t=0

1	2	0	2	3
2	2	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=1

1	2	0	2	3
2	3	2	3	0
1	2	3	3	2
3	1	3	2	1
0	2	2	1	2

t=2

1	2	0	2	3
2	3	2	3	0
1	2	4	3	2
3	1	3	2	1
0	2	2	1	2

t=3

1	2	0	2	3
2	3	3	3	0
1	3	0	4	2
3	1	4	2	1
0	2	2	1	2

t=4

1	2	0	2	3
2	3	3	4	0
1	3	2	0	3
3	2	0	4	1
0	2	3	1	2

t=5

1	2	0	3	3
2	3	4	0	1
1	3	2	2	3
3	2	1	0	2
0	2	3	2	2

t=6

1	2	1	3	3
2	4	0	1	1
1	3	3	2	3
3	2	1	0	2
0	2	3	2	2

t=7

1	3	1	3	3
3	0	1	1	1
1	4	3	2	3
3	2	1	0	2
0	2	3	2	2

t=8

1	3	1	3	3
3	1	1	1	1
2	0	4	2	3
3	3	1	0	2
0	2	3	2	2

t=9

1	3	1	3	3
3	1	2	1	1
2	1	0	3	3
3	3	2	0	2
0	2	3	2	2

Avalanche area

1	3	1	3	3
3	1	2	1	1
2	1	0	3	3
3	3	2	0	2
0	2	3	2	2

Adapted from:
P. Bak 96,
How nature Works,
The science of self-
organized criticality

Gutenberg-Richter Law (1949, 1954)

In part taken from “http://en.wikipedia.org/wiki/Gutenberg-Richter_law”, 5/2013

Gutenberg-Richter Law (1949, 1954)

The Gutenberg Richter law yields a statistical dependency of the number N of earthquakes with at least magnitude M , and the magnitude M (strength in logarithmic scale),

b is a constant value, which is typically close to $b=1.0$

Values of $0.5 < b < 1.5$ are reasonable in special environments.

In part taken from "http://en.wikipedia.org/wiki/Gutenberg-Richter_law", 5/2013

Gutenberg-Richter Law (1949, 1954)

The Gutenberg Richter law yields a statistical dependency of the number N of earthquakes with at least magnitude M , and the magnitude M (strength in logarithmic scale),

b is a constant value, which is typically close to $b=1.0$

Values of $0.5 < b < 1.5$ are reasonable in special environments.

$$\log_{10}(N) = a - bM$$

$$N = 10^{a-bM}$$

In part taken from "http://en.wikipedia.org/wiki/Gutenberg-Richter_law", 5/2013

Zipf's Law (1935, 1949)

Zipf's law is a statistical observation relating the frequency of a word in a data set to the rank of this word in the sorted list.

In part taken from “http://en.wikipedia.org/wiki/Zipf's_law”, 5/2013

Zipf's Law (1935, 1949)

Zipf's law is a statistical observation relating the frequency of a word in a data set to the rank of this word in the sorted list.

Zipf's law states that given some corpus of natural language utterances, the frequency $f(r)$ of any word is inversely proportional to its rank r in the frequency table.

Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.

In part taken from "http://en.wikipedia.org/wiki/Zipf's_law", 5/2013

Zipf's Law (1935, 1949)

Zipf's law is a statistical observation relating the frequency of a word in a data set to the rank of this word in the sorted list.

Zipf's law states that given some corpus of natural language utterances, the frequency $f(r)$ of any word is inversely proportional to its rank r in the frequency table.

Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.

$$f(r) \sim \frac{1}{r^\gamma}$$

$$f(r, \gamma, N) = \frac{1}{r^\gamma} * \frac{1}{\sum_{n=1}^N 1/n^\gamma}$$

N: no of words, size of corpus
 r: rank of word in sorted list
 γ : characteristic exponent ,
 very often: $\gamma \approx 1$

In part taken from "http://en.wikipedia.org/wiki/Zipf's_law", 5/2013

Zipf-Mandelbrot Law (1935, 1949)

An extension to Zipf's law is the Zipf-Mandelbrot Law, with a slightly modified relation between word rank and word frequency.

The special case that $b=0$ yields exactly Zipf's Law.

$$f(r) \sim \frac{1}{(r+b)^{\gamma}}$$

Typical values that have been found are:

$\gamma_Z=1.01$ and $\gamma_{ZM}=1.06$, $b_{ZM}=1.17$ (British National Corpus BNC, 1997)

$\gamma_Z=0.97$ and $\gamma_{ZM}=1.03$, $b_{ZM}=2.29$ (DeReKo German Reference Kopus, 2011)

Lotka's Law (1926)

Lotka's law is in principle the same observation as Zipf's Law. It describes the frequency of publications by authors in any given field. As the number of articles published increases, authors producing that many publications become less frequent.

It states, that the number $Y(x)$ of authors making x contributions in a given period, is a fraction of the number making a single contribution, following the formula:

$$Y(x) = \frac{C}{x^a}$$

where $a \approx 2.0$ an approximate inverse-square law, where the number of authors publishing a certain number of articles is a fixed ratio to the number of authors publishing a single article.

There are 1/4 as many authors publishing 2 articles within a specified time period as there are single-publication authors,

1/9 as many publishing 3 articles, 1/16 as many publishing 4 articles, etc.

In part taken from "http://en.wikipedia.org/wiki/Lotka's_law", 5/2013

Auerbach's Detection (1913), Pareto Distribution

The relationship between the rank (by size) of a city, and the population is following the same power law as Zipf's law.

In 1913 Felix Auerbach discovered, that the rank of the cities, sorted by population size and the population of these cities is related to each other following a power law.

This has been re-found later in 1953 and 1993 by other researchers.

In fact, the same principle relation for continuous distributions has been published already in 1897 by *Vilfredo Pareto*.

The *Pareto distribution* has colloquially become known and referred to as the *Pareto principle*, or the "80-20 rule".

This rule states that,
for example, 80% of the wealth of a society is held by 20% of its population,
or that 80% of the information can be obtained by 20% of the sources.

Auerbach's Detection

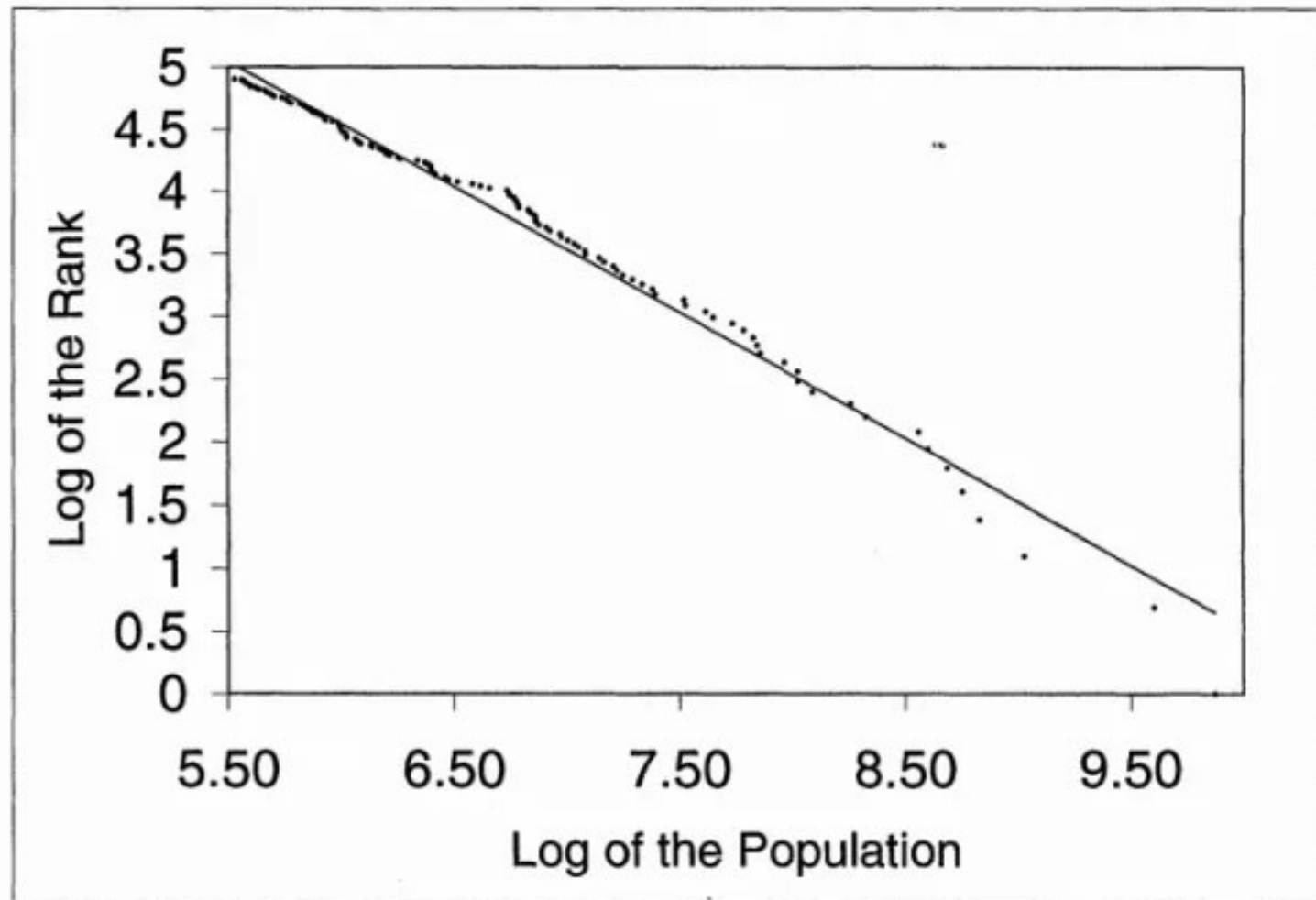


FIGURE I

Log Size versus Log Rank of the 135 largest U. S. Metropolitan Areas in 1991
Source: Statistical Abstract of the United States [1993].

Examples of SOC Systems

- **Forest fire model** (K. Chen, P. Bak, M. H. Jensen: 1990
B. Drossel, F. Schwabl: 1992)
- **Sandpile model** (P. Bak, C. Tang, K. Wiesenfeld: 1987)
- **Land slides** (Y. Fuii: 1969)
- **Percolation Theory** (S. Broadbent, J. Hammersley, 1957)
- **Earthquakes** (B. Gutenberg, C.F. Richter, 1949, 1954)
- **Zipf's Law** (G.K. Zipf: 1935)
- **Lotka's Law** (A.J. Lotka: 1926)
- **Auerbach's Detection** (F. Auerbach, 1913)

Topics

- Self Organized Criticality, SOC
- **Ant Algorithms**

Historic Remarks, Different Approaches

Evolutionary Computation (EC)

Swarm Behavior / Swarm Intelligence

- **Ant Algorithm**
- **Ant Colony Optimization**
- Particle Swarm Optimization

Evolutionary Algorithms (EA)

- Genetic Algorithms (GA)
- Genetic Programming (GP)
- Evolutionary Strategies (ES)
- Evolutionary Programming (EP)

Ant Algorithms, Ant colony optimization

Ant Algorithm

The [Ant Algorithm](#) is a method for discrete optimization. It has been inspired by observations of typical behaviors of ants, and colonies of ants.

Since the first publications on [Ant Algorithms](#) by M.Dorigo and colleagues in 1991 and 1992, a lot of further developments, enhancements and applications haven been proposed, and published.

Meanwhile a community of researchers and research groups on [Ant Algorithms](#) has been established.

M.Dorigo: *Optimization, Learning and Natural Algorithms*,
PhDThesis, (in Italian) Dipartimento die Elletronica, Politecnico di Milano, IT, 1992

M.Dorigo, V.Maniezzo, and A.Coloni. *Positive feedback as a search strategy*.
Technical Report 91-016, Dipartimento die Elletronica, Politecnico di Milano, IT, 1991

Ant Algorithms

“Ant Algorithms were inspired by the observation of real ant colonies.

Ants are social insects, that is, insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole than to that of a single individual component of the colony.

Social insects have captured the attention of many scientists because of the high structuration level their colonies can achieve, especially when compared to the relative simplicity of the colony's individuals.

An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find shortest paths between food sources and their nest.”

Text taken from: Marco Dorigo and Gianni Di Caro, Luca M. Gambardella:
Ant Algorithms for Discrete Optimization Tech. Report IRIDIA / 98-10

Ant Algorithm

The main idea behind the [Ant Algorithm](#) is to have a system of cooperating agents, that can reach a solution in a better, or more efficient way, than a single individual would do.

Essential ingredients of an Ant System (AS) are:

- Multiple cooperating agents,
- which are simple structured;
- they have a sensory system,
- a method to deposit pheromones (stigmergy),
- and a simple mechanism to decide where to go.
- The pheromones evaporate after a while.

Ant Algorithm

Ants are leaving the nest N searching for food.
They leave the nest heading into arbitrary directions.

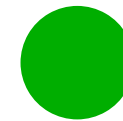
Ant Algorithm

Ants are leaving the nest N searching for food.
They leave the nest heading into arbitrary directions.

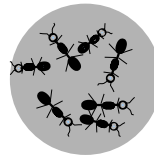
During their journey they will deposit pheromones along their pathway.

Ant Algorithm

A nest, some ants, and a pile of food.



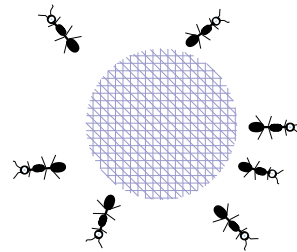
Some food.



A nest of ants.

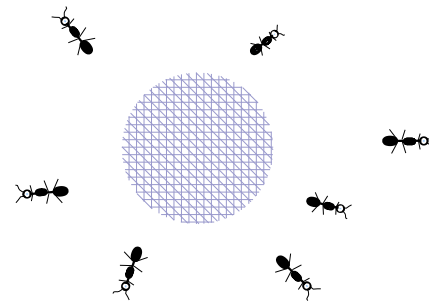
Ant Algorithm

The ants are leaving the nest ...



Ant Algorithm

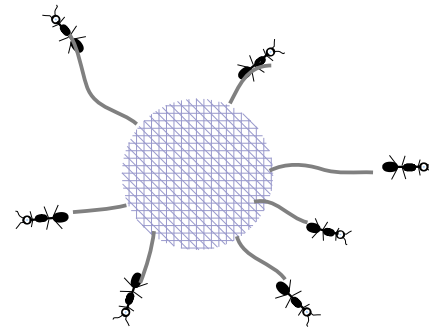
The ants are leaving the nest ...
... heading into arbitrary directions.



Ant Algorithm

The ants are leaving the nest ...
... heading into arbitrary directions.

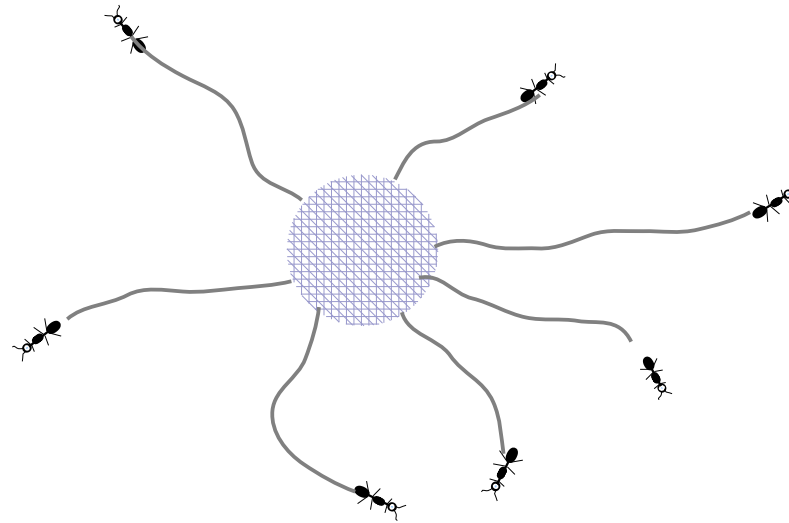
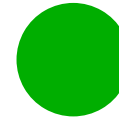
They drop pheromones along the trail.



Ant Algorithm

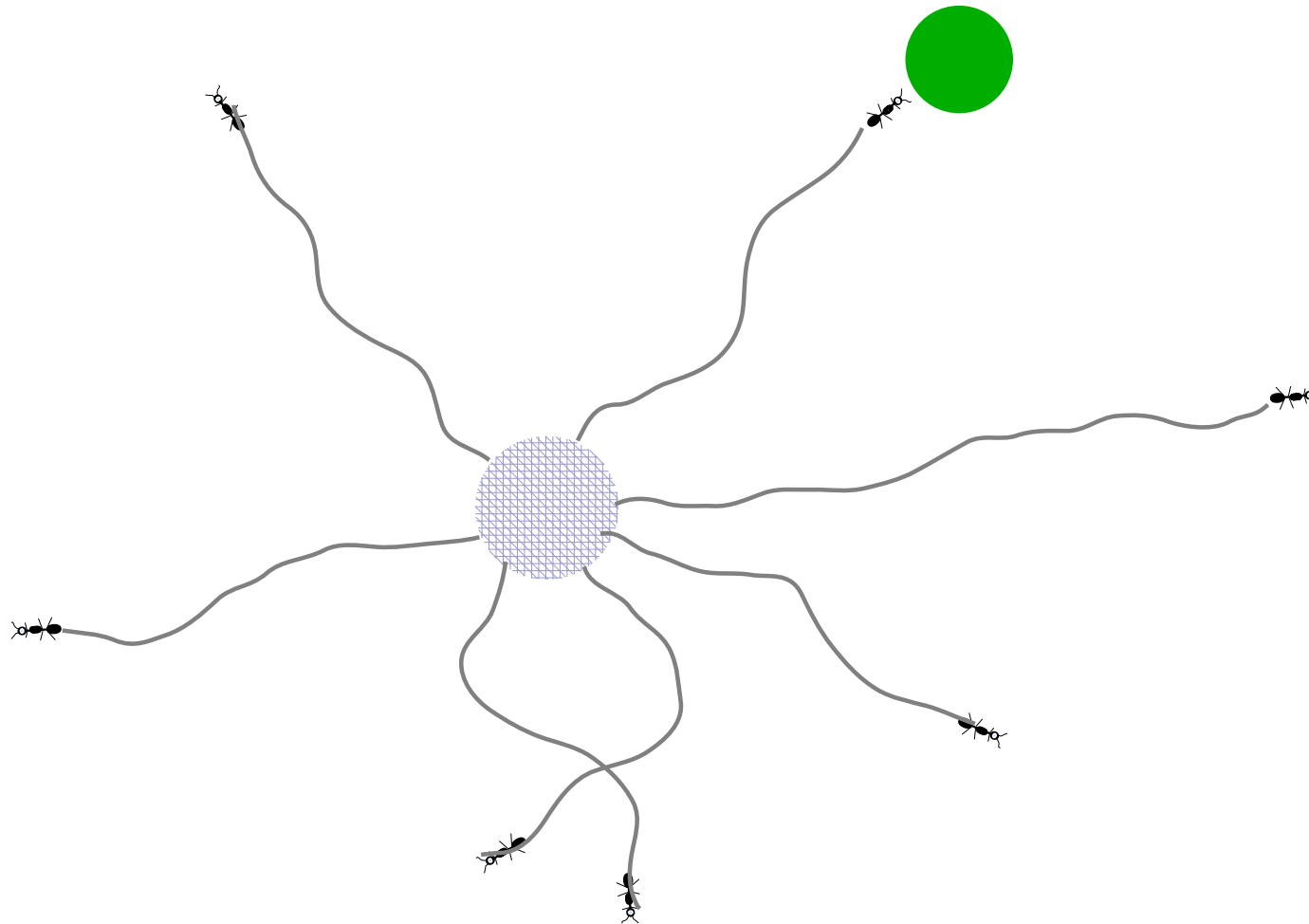
The ants are leaving the nest ...
... heading into arbitrary directions.

They drop pheromones along the trail.



Ant Algorithm

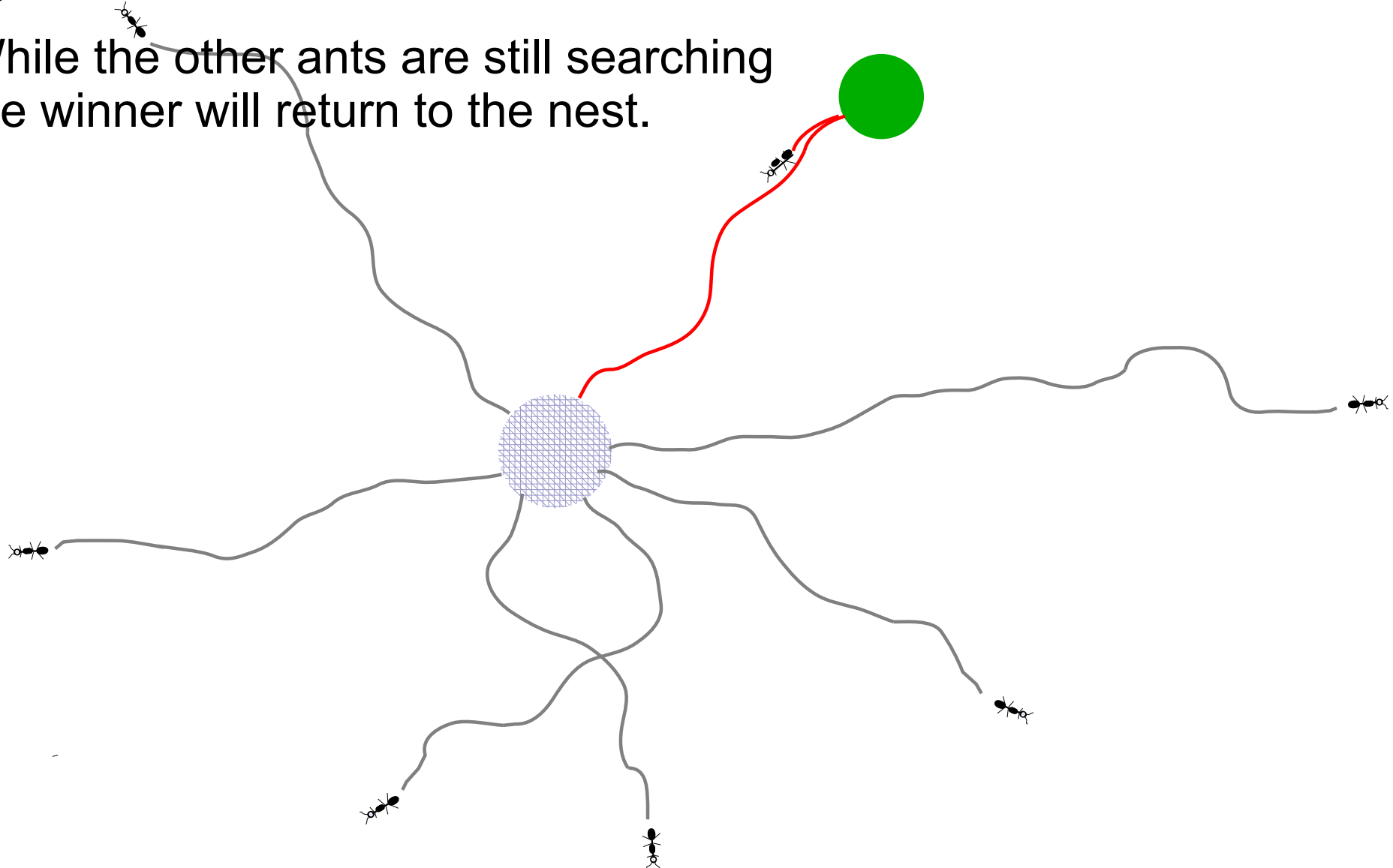
By chance, one of the ants has reached the source of food.



Ant Algorithm

By chance, one of the ants has reached the source of food.

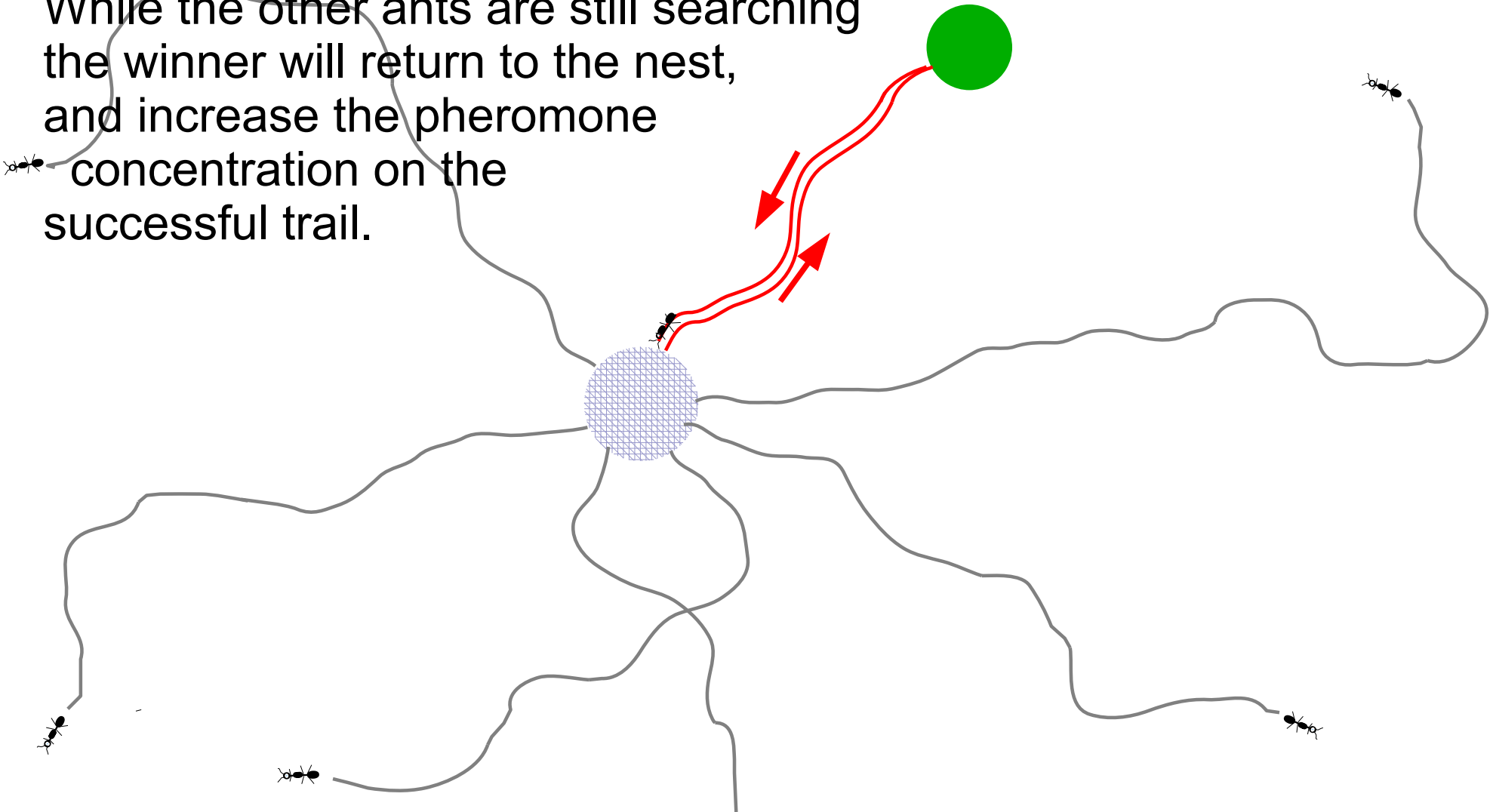
While the other ants are still searching
the winner will return to the nest.



Ant Algorithm

By chance, one of the ants has reached the source of food.

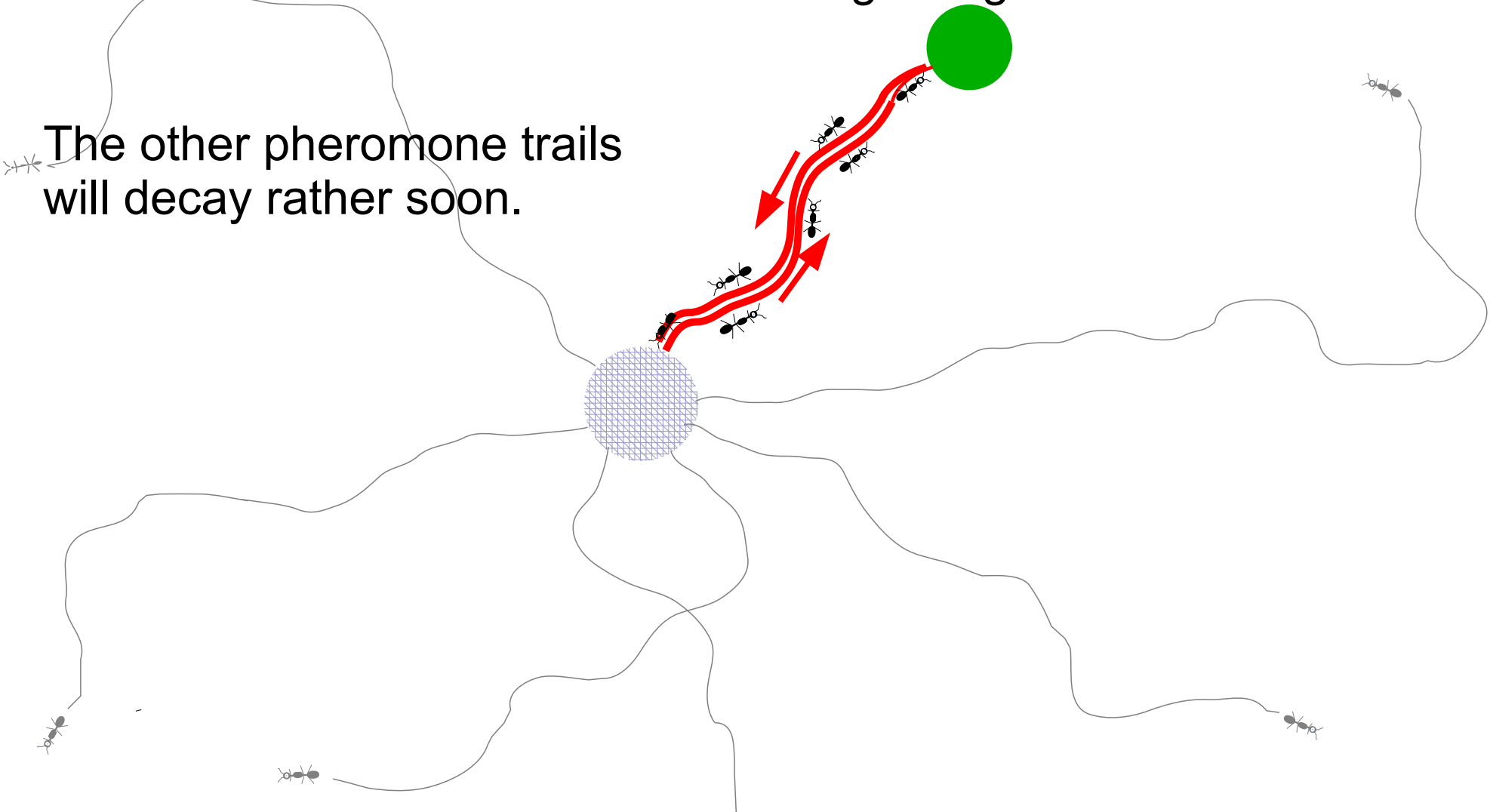
While the other ants are still searching the winner will return to the nest, and increase the pheromone concentration on the successful trail.



Ant Algorithm

The pheromone concentration on the successful path will rise, while more and more ants are traveling along this trail.

The other pheromone trails will decay rather soon.



Ant Algorithms

Ants are leaving the nest N searching for food.
They leave the nest heading into arbitrary directions.

During their journey they will deposit pheromones along their pathway.

Eventually one will reach an area where food is located.
There, it will get some food, and will then travel back to the nest.
The ant is following it's own pheromone trail, and will thus (almost) take the same route back.

Even during the return travel, the ant will drop some pheromone, thus increasing the concentration of pheromones along the path.

After reaching the nest, the pheromone trail will guide this ant, and other ants to the food area.

Ant Algorithm

Since the pheromones will evaporate over time, and the trail that has reached a food source first, will be taken twice (forth and back), and thus have a higher pheromone concentration compared to the trails of the other ants.

The ants can sense the pheromone, and they base their movement on the pheromone concentration.

The higher the pheromone concentration is, the more likely an ant will take the respective path, or instead will follow an arbitrary route (almost random).

Other ants will base their journey on the pheromone concentration. Thus, the most successful path will get a higher probability to be chosen, more ants will take this path, and will further increase the pheromone concentration (positive feedback, autocatalytic mechanism).

Ant Algorithm

Ants, that do not have any prior pheromone trail, are just performing random movements.

Random movements are a method to find a path ; although this is slow and inefficient, it can be shown that there is a chance for even finding the optimal path.

Ant Algorithm

Ants, that do not have any prior pheromone trail, are just performing random movements.

Random movements are a method to find a path ; although this is slow and inefficient, it can be shown that there is a chance for even finding the optimal path.

Ants, that follow a pheromone trail are using the knowledge that has been acquired before.

The more successful a path has been before, the more likely it will be taken. Those trails are local minima of the underlying optimization problem.

Ant Algorithm

Ants, that do not have any prior pheromone trail, are just performing random movements.

Random movements are a method to find a path ; although this is slow and inefficient, it can be shown that there is a chance for even finding the optimal path.

Ants, that follow a pheromone trail are using the knowledge that has been acquired before.

The more successful a path has been before, the more likely it will be taken. Those trails are local minima of the underlying optimization problem.

The process of time dependent evaporation is tricky:

- it enforces faster, and thereby shorter routes,
- it can react on dynamic changes of the environment, exhausted source of food, changing path length, ...
- virtual reset of unused trails.

Ant Algorithm

Ants, that do not have any prior pheromone trail, are just performing random movements: **Exploration**.

Random movements are a method to find a path ; although this is slow and inefficient, it can be shown that there is a chance for even finding the optimal path.

Ants, that follow a pheromone trail are using the knowledge that has been acquired before: **Exploitation**.

The more successful a path has been before, the more likely it will be taken. Those trails are local minima of the underlying optimization problem.

The process of time dependent evaporation is tricky:

- it enforces **faster**, and thereby **shorter routes**,
- it can react on **dynamic changes** of the environment, exhausted source of food, changing path length, ...
- **virtual reset** of unused trails.

Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.

Just make the ants travel along a graph, where the starting point (nest) and the goal (food) are two special nodes, the other nodes are connected by edges representing the allowed routes to travel.

Within each discrete time step an ant can travel one edge, the pheromones are disposed on the edges reciprocal to the length of the edge.

Evaporation of the pheromones, is implemented by an exponential decay.

The pheromone dependent decision is implemented using a softmax, or a Boltzmann distribution.

Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.



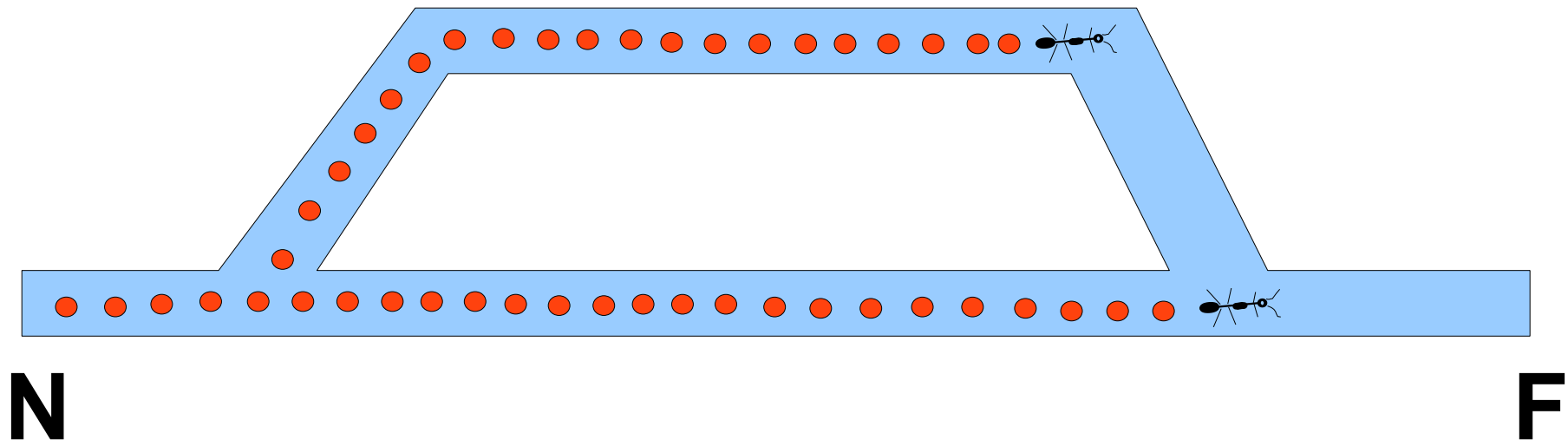
Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.



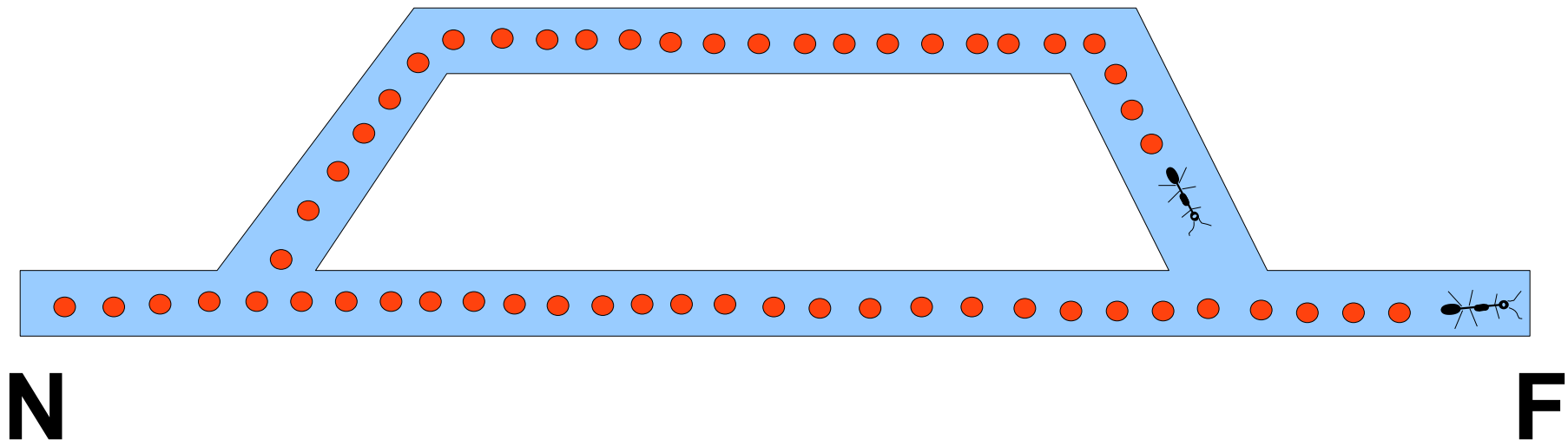
Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.



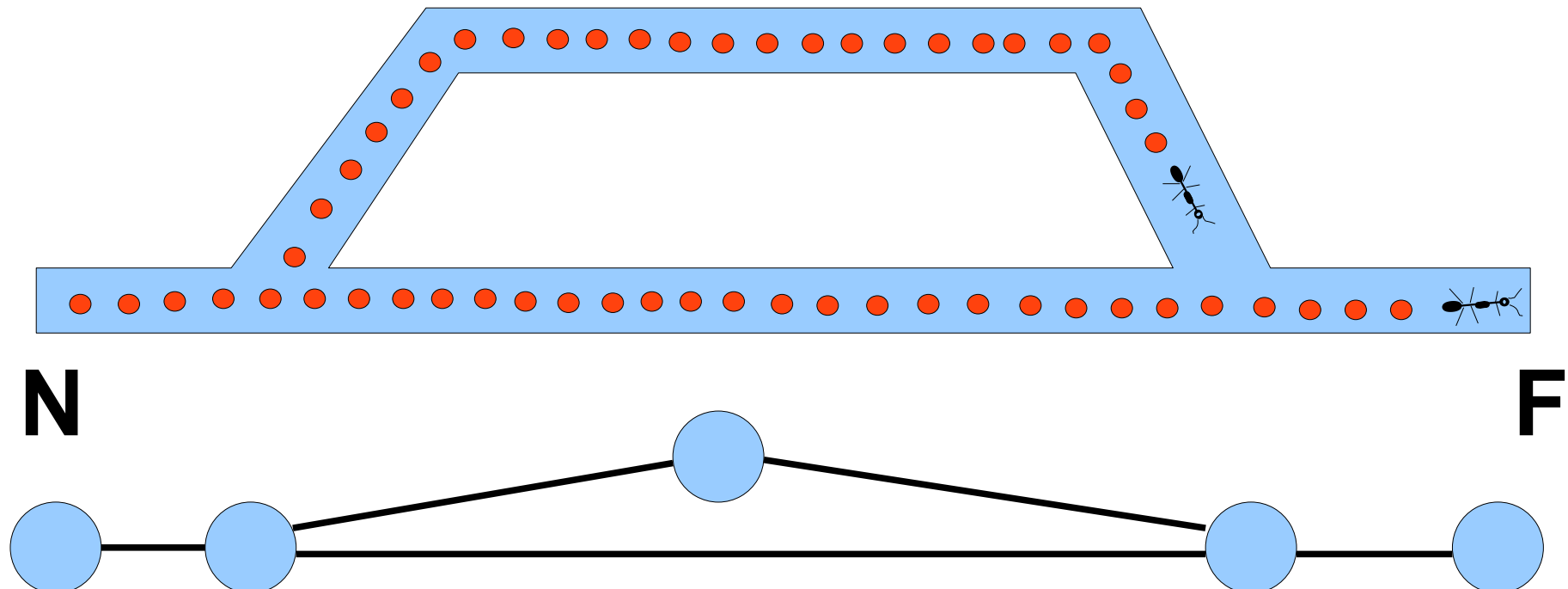
Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.



Ant Algorithms

Transposing the ant algorithm, from real valued trails, to a discrete world is straight forward.



Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

Ant Colony System, ACS:

Ant Colony Optimization, ACO:

AntNet:

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

The pheromone update is only performed after the journey of an ant has been completed (e.g. nest \rightarrow food \rightarrow nest).

Ant Colony System, ACS:

Ant Colony Optimization, ACO:

AntNet:

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

The pheromone update is only performed after the journey of an ant has been completed (e.g. nest \rightarrow food \rightarrow nest).

Ant Colony System, ACS:

Like AS, but the ant decision is a greedy decision, and, only the best ant will generate a pheromone trail.

Ant Colony Optimization, ACO:

AntNet:

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

The pheromone update is only performed after the journey of an ant has been completed (e.g. nest \rightarrow food \rightarrow nest).

Ant Colony System, ACS:

Like AS, but the ant decision is a greedy decision, and, only the best ant will generate a pheromone trail.

Ant Colony Optimization, ACO:

Meta-heuristics comprising AS, and ACS methods, but with further options to change decision and pheromone dropping.

AntNet:

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

The pheromone update is only performed after the journey of an ant has been completed (e.g. nest \rightarrow food \rightarrow nest).

Ant Colony System, ACS:

Like AS, but the ant decision is a greedy decision, and, only the best ant will generate a pheromone trail.

Ant Colony Optimization, ACO:

Meta-heuristics comprising AS, and ACS methods, but with further options to change decision and pheromone dropping.

AntNet:

An application of ACO for finding an adequate network-routing.

Ant Algorithms

Beside the original Ant Algorithm, a set of variants have been proposed. They differ in the way, the pheromone trail is updated, and how the ants decide.

Ant System, AS:

The pheromone update is only performed after the journey of an ant has been completed (e.g. nest \rightarrow food \rightarrow nest).

Ant Colony System, ACS:

Like AS, but the ant decision is a greedy decision, and, only the best ant will generate a pheromone trail.

Ant Colony Optimization, ACO:

Meta-heuristics comprising AS, and ACS methods, but with further options to change decision and pheromone dropping.

AntNet: (G.Di Caro, M.Dorigo, (1998), <http://arxiv.org/pdf/1105.5449>)

An application of ACO for finding an adequate network-routing.

(AntNet: Distributed Stigmergetic Control for Communications Networks)

Topics

- Self Organized Criticality, SOC
- Ant Algorithms

Artificial Life Summer 2025

Self Organized Criticality, SOC Ant Algorithms

Master Computer Science [MA-INF 4201]

Mon 14:15 – 15:45, HSZ, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Artificial Life Summer 2025

Self Organized Criticality, SOC
Ant Algorithms

Thank you for your patience

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn