# Artificial Life
# Summer 2015

## Self Organized Criticality, SOC
## Ant Algorithms

Master Computer Science [MA-INF 4201]
Mon 8:30 – 10:00, LBH, Lecture Hall III.03a

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






## Topics

- **Self Organized Criticality, SOC**
- Ant Algorithms
- Some basics on optimization






## Overview: SOC

- What is Self Organized Criticality?
- Motivation
- Power law , Scaling law
- Examples of SOC systems
  - Sandpile model
  - Land slides
  - Forest fire model
  - Earthquakes
  - Zipf's Law






## 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.

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**.

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 and 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.





## 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?

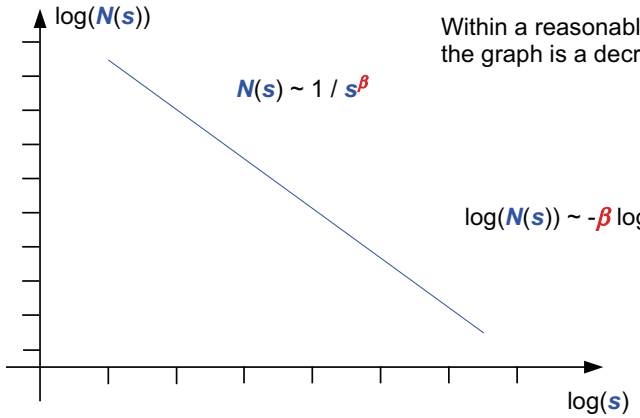
$$N(s) \sim 1 / s^{\beta}$$

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

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.




Within a reasonable limit for ***s*** and ***N***, the graph is a decreasing line.

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





## 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 $\tau$ between events,
The Inter-event-interval distribution

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

### 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

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





## 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 **T**ree
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.

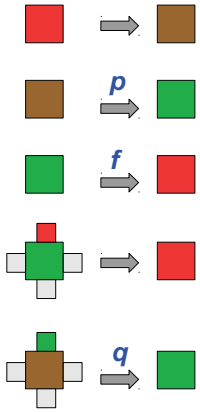
Fire turns into ashes
p — Spontaneous growth
f — Spontaneous fire
Induced fire: a tree burns if at least one neighbor burns
q — Induced growth: a tree grows if at least one neighbor is a tree

# 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 mircro-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 << p* (e.g. *p/f* = 100).



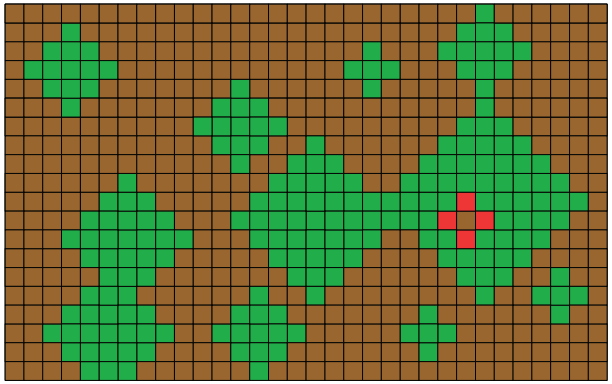


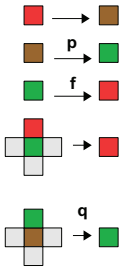




# SOC example: Forest-Fire Model

**Forest-Fire CA**
d=2, rectangular grid, r=1, von Neumann, k=3
**non deterministic** cellular automaton,






q = 1.0






# Gutenberg-Richter Law (1949, 1954)

The Gutenberg Richter law yields a statistical dependency of the number ***N*** of earthquakes with magnitude ≤ M, and the magnitude ***M*** (strength in logarithmic scale),
*b* is a constant value 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.

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: γ ≈ 1

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

# Artificial Life
# Summer 2015

## Self Organized Criticality, SOC
## Ant Algorithms

Master Computer Science [MA-INF 4201]
Mon 8:30 – 10:00, LBH, Lecture Hall III.03a

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






## Topics

- Self Organized Criticality, SOC
- **Ant Algorithms**
- Some basics on optimization






## 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, Politechnico di Milano, IT, 1992

M.Dorigo, V.Maniezzo, and A.Colorni. *Positive feedback as a search strategy*.
Technical Report 91-016, Dipartimento die Elletronica, Politechnico 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

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

Some food.

A nest of ants.





# 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.

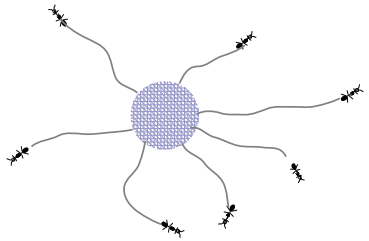




# Ant Algorithm

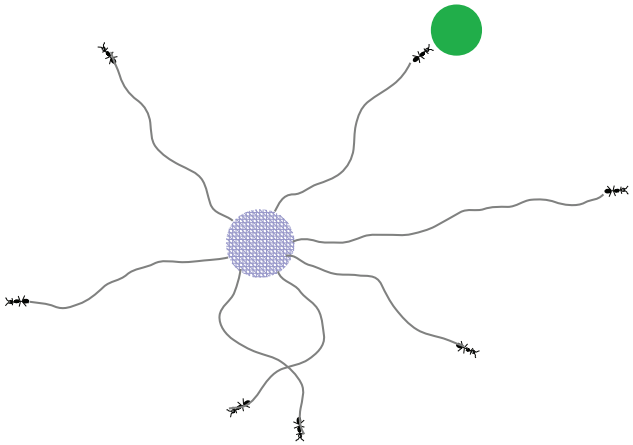
The ants are leaving the nest …
… heading into arbitrary directions.

They drop pheromones along the trail.



## Ant Algorithm

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



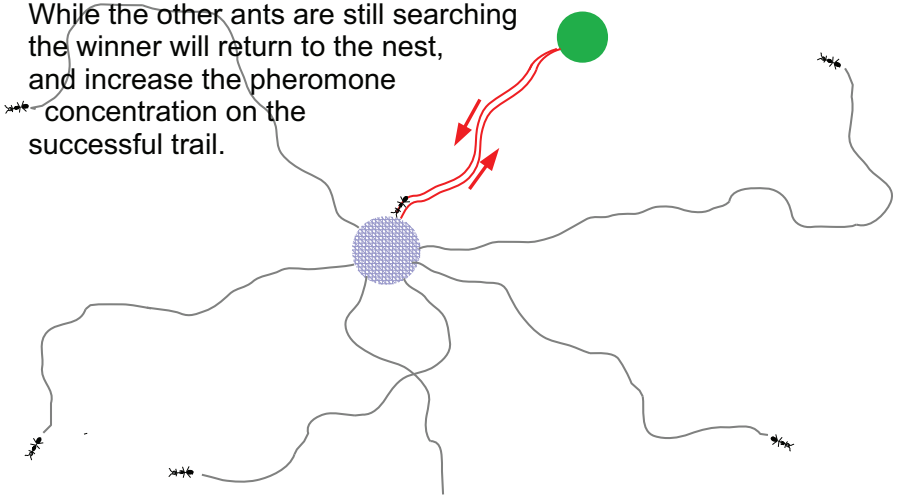




## Ant Algorithm

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, and increase the pheromone concentration on the successful trail.



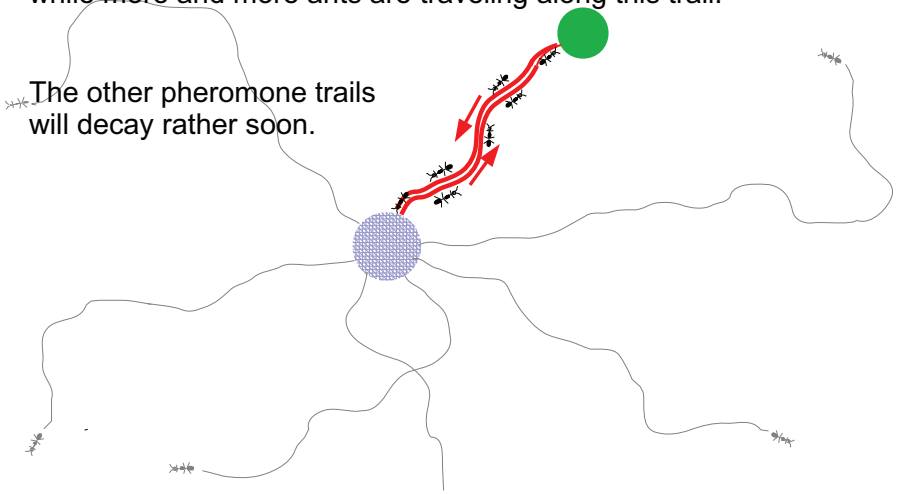




## Ant Algorithm

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 Algorithm

Since the pheromones will evaporate over time, the trail that has reached a food source first, will be taken twice (forth and back), and thus will 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: **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.

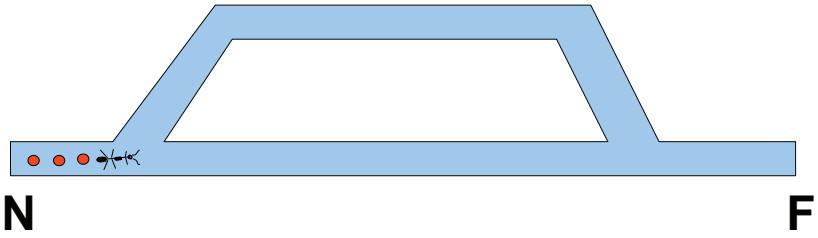
The pheromone dependent decision is implemented using a softmax, or a Boltzmann distribution.






# Ant Algorithms

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



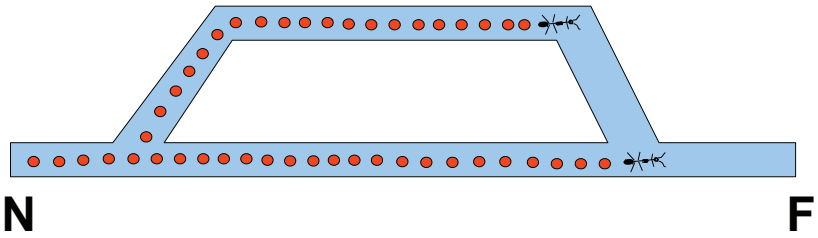






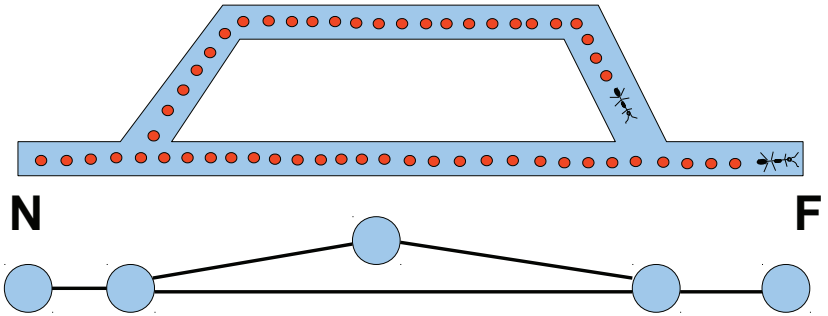
# Ant Algorithms

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

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 → food → 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
- **Some basics on optimization**





## Optimization (some basics):

Depending on the background and the context, optimization is mainly seen as a part of :
- Operations Research
- Artificial Intelligence, or more precise
- Computational Intelligence.

Some Artificial Life approaches are directly working in the following subsets of optimization:
- Stochastic Optimization
- Meta heuristic Optimization

## Optimization (some basics):

(Global) optimization is typically based on optimizing a criterion of some objective function **f(s)**:
- minimizing some costs
- maximizing some fitness

Optimization is the process of finding a position **s*** in N-dimensional search space **S** that optimizes the objective function **f(s)** $\in \mathbb{R}$,

**f(s*) ≤ f(s)** $\forall$ s$\in$ **S** (in case of minimization)

**f(s*) ≥ f(s)** $\forall$ s$\in$ **S** (in case of maximization)





## Minimization vs. Maximization:

**Minimization** is a kind of optimization with respect to (scalar) **costs**. Costs are typically defined to have a known lower bound: usually „zero“.

**Maximization** is a kind of optimization with respect to a (scalar) objective function; like **fitness**, profit, or performance. A realistic upper bound for this objective function is often not known in advance.

Re-formulating a **maximization** problem into a **minimization** problem is often possible, but not always advisable because of a probable non-linear complete re-scaling of the objective function.





## Global vs. Local Optimization:

Global optimization attempts to find the absolute best (global) optimum for a given problem.
Finding a global optimum (there might be several) is definitively one of the the hardest tasks in optimization.

Beside the global optimum, there can be local optima (typically there are a lot of them).

Local optimization is attempting to find an extremum in the vicinity of a starting point, that is good, or at least sufficient for the given problem.
There is a chance to find a global optimum, even with local optimization methods; **but you never know.**





## Stochastic Optimization:

Stochastic optimization is including a stochastic component into the optimization process. The stochastic component can be part of every aspect of the process.

In can be:
- in the data,
- in the objective function,
- in the optimization method as explicit component,
- in the schedule of the optimization process.

The stochastic component can be an inevitable property of the application one has to cope with,
or can be included deliberately to support the algorithm.

## Stochastic Optimization:

There is a variety of other approaches from the field of Stochastic optimization:

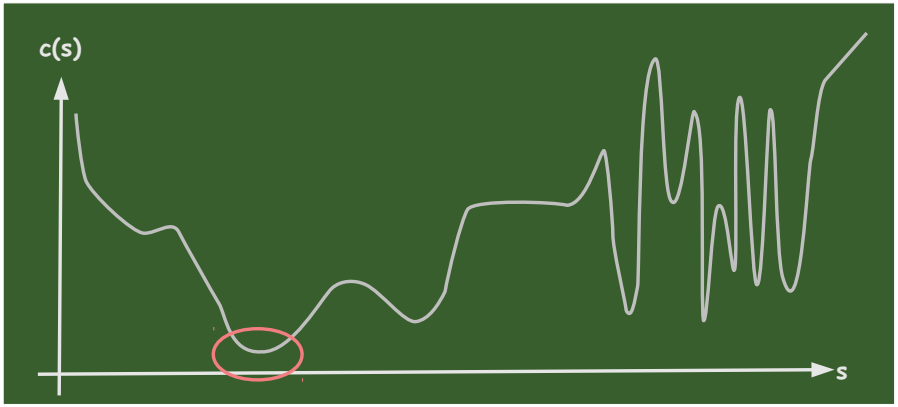
- Random Search
- Random Optimization
- Monte Carlo Methods
- Simulated Annealing
- Single Start
- Multi Start
- Random Search Direction Methods
- Hill Climbing
- …





## Optimization:

Draw a **nasty cost function** on the blackboard:

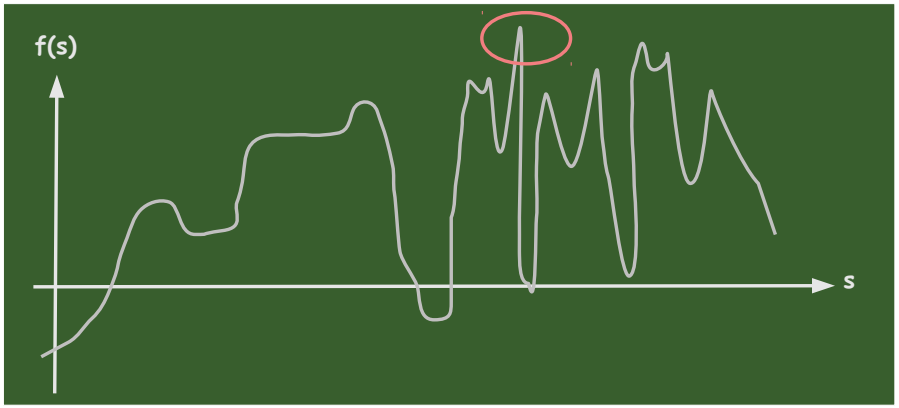







## Optimization:

Draw a **nasty fitness function** on the blackboard:








## Optimization:

A typical **playground** to investigate the capabilities of optimization methods are **test functions** to be minimized.

Over the years, many researchers have invested a lot of time to design adequate and appealing test functions to demonstrate and evaluate the results of their optimization methods.

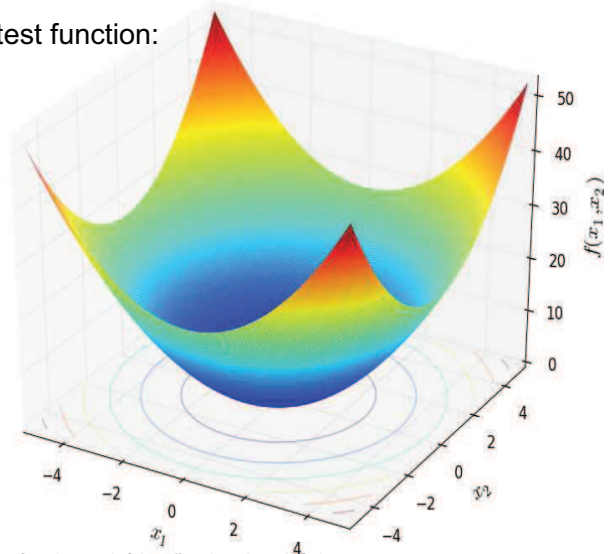
Meanwhile, many collections of such functions can be found.
A nice collections is:

http://infinity77.net/global_optimization/test_functions.html#test-functions-index

# Optimization:

An easy to understand test function:

$$f_{\text{Sphere}}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$$



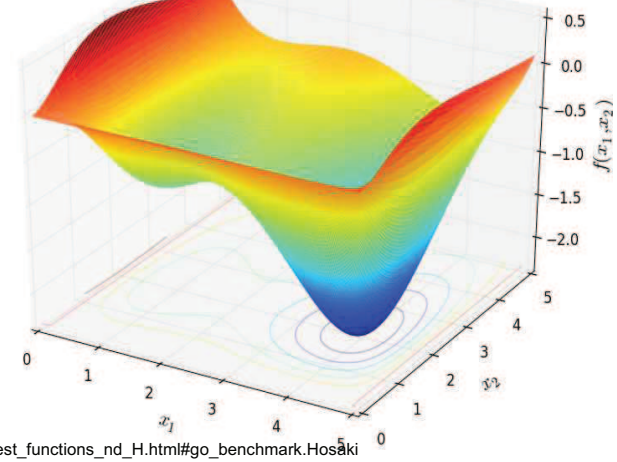

http://infinity77.net/global_optimization/test_functions_nd_S.html#go_benchmark.Sphere





# Optimization:

$$f_{\text{Hosaki}}(\mathbf{x}) = \left(1 - 8x_1 + 7x_1^2 - \frac{7}{3}x_1^3 + \frac{1}{4}x_1^4\right) x_2^2 e^{-x_1}$$



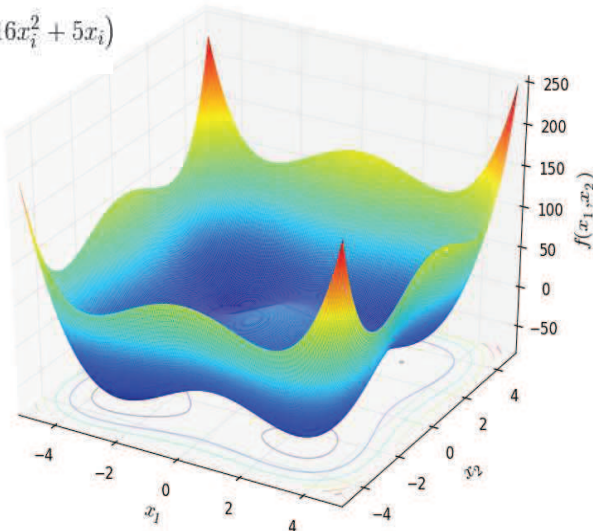

http://infinity77.net/global_optimization/test_functions_nd_H.html#go_benchmark.Hosaki





# Optimization:

$$f_{\text{StyblinskiTang}}(\mathbf{x}) = \sum_{i=1}^{n} \left(x_i^4 - 16x_i^2 + 5x_i\right)$$



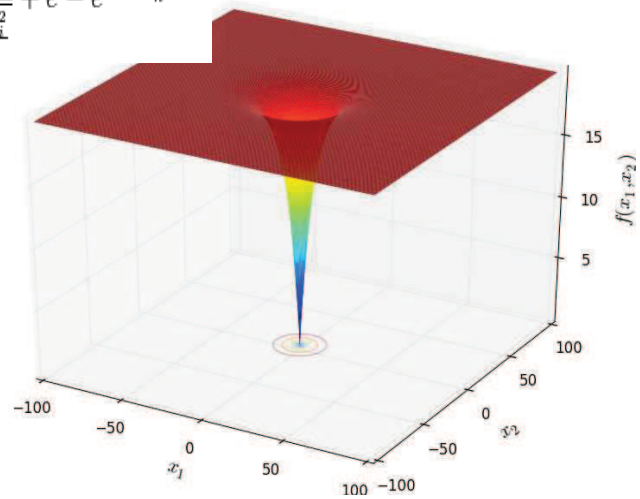

http://infinity77.net/global_optimization/test_functions_nd_S.html#go_benchmark.StyblinskiTang





# Optimization:

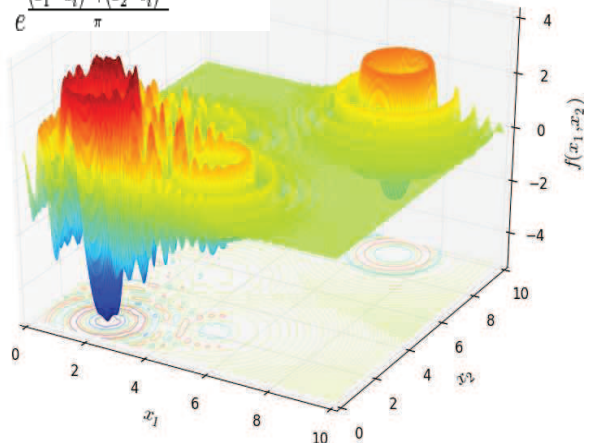
$$f_{\text{Easom}}(\mathbf{x}) = a - \frac{a}{e^{b\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}}} + e - e^{\frac{\sum_{i=1}^{n} \cos(cx_i)}{n}}$$




http://infinity77.net/global_optimization/test_functions_nd_E.html#go_benchmark.Easom

## Optimization:

$$f_{\text{Langermann}}(\mathbf{x}) = -\sum_{i=1}^{5} \frac{c_i \cos\left\{\pi\left[(x_1-a_i)^2+(x_2-b_i)^2\right]\right\}}{e^{\frac{(x_1-a_i)^2+(x_2-b_i)^2}{\pi}}}$$



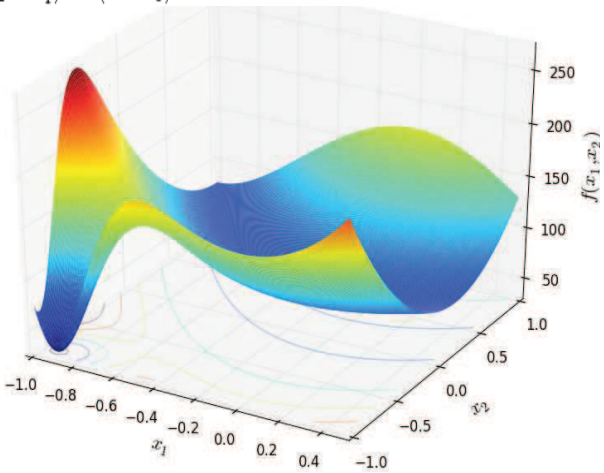

http://infinity77.net/global_optimization/test_functions_nd_L.html#go_benchmark.Langermann





## Optimization:

$$f_{\text{RosenbrockModified}}(\mathbf{x}) = 74 + 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2 - 400e^{-\frac{(x_1+1)^2+(x_2+1)^2}{0.1}}$$



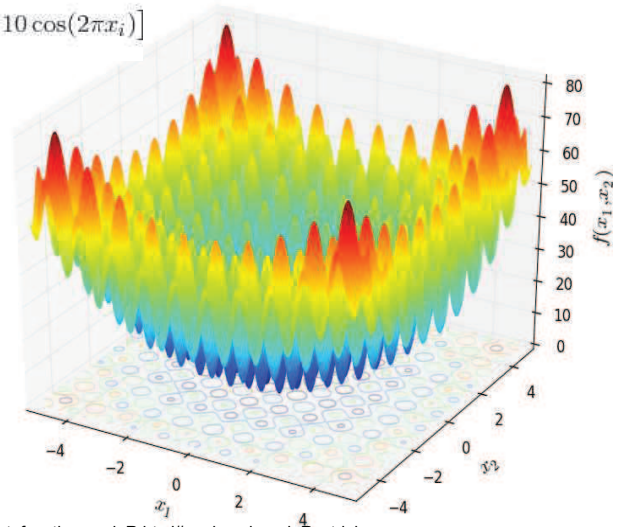

http://infinity77.net/global_optimization/test_functions_nd_R.html#go_benchmark.RosenbrockModified





## Optimization:

$$f_{\text{Rastrigin}}(\mathbf{x}) = 10n \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$



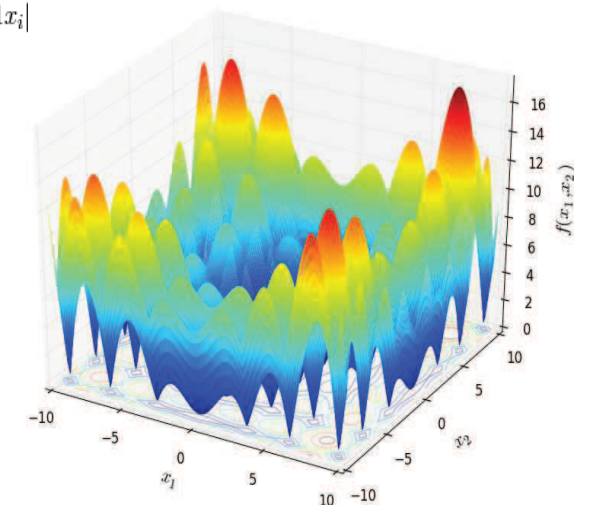

http://infinity77.net/global_optimization/test_functions_nd_R.html#go_benchmark.Rastrigin





## Optimization:

$$f_{\text{Alpine01}}(\mathbf{x}) = \sum_{i=1}^{n} \left| x_i \sin\left(x_i\right) + 0.1x_i \right|$$




http://infinity77.net/global_optimization/test_functions_nd_A.html#go_benchmark.Alpine01

## Optimization:

Below is a list of common **test functions (to minimize)** that are used to investigate the capabilities of optimization methods:

- Schwefel's Function
- Generalized Rosenbrock's Function
- Rastrigin's Function
- Ackley's (path) Function
- Langermann's Function
- Michalewicz's Function
- Easoms Function
- Griewangk's Function
- Bohachevsky's Function
- Watson's Function
- Colville's Function
- …





## Some important dates:

**This Wednesday, 20.5.15, is a special day at University of Bonn**
**dies academicus**

There are no regular lectures after 10:00 that day,
but a lot of public talks, most at the central University Building.
http://www3.uni-bonn.de/studium/studium-universale/dies-academicus/dies-zeitung/view

**Next weekend, Sun 24.5.15 – Mon 25.5.15 is Pentecost/Whitsun**

**Thus, Monday 25.5.15 is Holiday in Germany**
Shops are closed, University will be closed.

**Next week: Tue 26.5.15 – Fri 29.5.15 is a "*no teaching week*"**

There will be **no lectures**, **no exercise groups**, …
but University buildings will be open.

Next **Artificial Life** lecture is: **Mon 1.6.2015**





## Some important dates:

Written examination, is scheduled for:
Thursday, **30. July 2015** from **10:00** to 11:40,
LBH Building, Lecture Hall: III.03a

Re-Sit examination, will be:
Tuesday, **8. Sept 2015** from **10:00** to 11:40,
LBH Building





# Artificial Life
# Summer 2015

## Self Organized Criticality, SOC
## Ant Algorithms

**Thank you for your patience**

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