

Gemfony  scientific

**Distributed Parametric Optimization  
with the  
Geneva Library Collection**

Dr. Rüdiger Berlich

contact@gemfony.com | <http://www.gemfony.com>

BoostCon 2011 / Aspen, CO / May 16, 2011

# Karlsruhe Institute of Technology (North Campus)



Steinbuch Centre for  
Computing (SCC)



- Start-Up
  - A spin-off from KIT
- Founders with many years experience in
  - Modelling complex systems
  - Optimization
  - Cloud- and Grid-Management
  - IT-Security
- Main product: Geneva
  - **Grid-enabled evolutionary algorithms**
  - Accelerated **parametric optimization** for problems from a wide array of problem domains, both in industry and science
  - Heavily based on Boost



Who in this room has  
heard before the term  
**parametric optimization**  
before this talk?

Who in this room has  
**used parametric optimization**  
to improve the results of his/her work ?

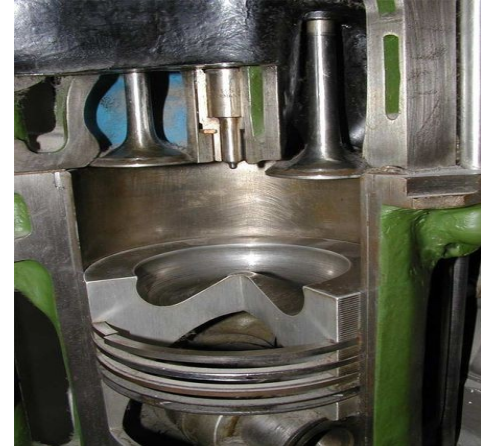
- Defining „parametric optimization“
- Some Use-Cases
- The Geneva Library Collection
  - Design decisions
  - Features
  - Libraries and Architecture
  - Performance
  - Status
- Experiences made with
  - Boost
  - Open Source

# Optimization Problems

Optimization problems can be found in just about every field of engineering, natural sciences as well as business and economic sciences (and every other part of life)

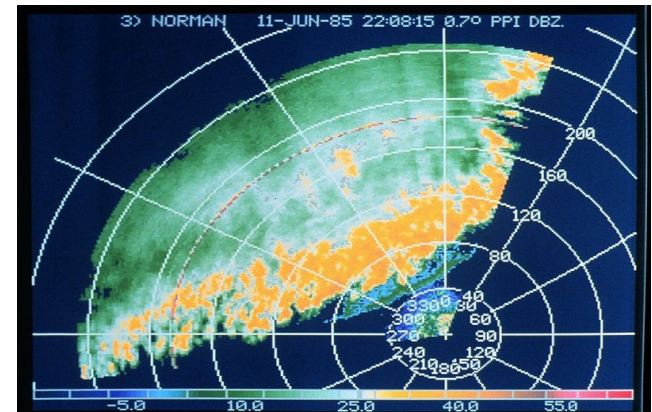
# Engineering and Simulations

- Optimization of combustion engines
  - Simultaneous adaption of large amounts of parameters
- Simulations
  - E.g. weather, traffic, ...
  - Callibration of „constant“ parameters, so a simulation does better predictions
  - E.g.: let the simulation „predict“ yesterday's weather based on the weather records stored for the past 100 years



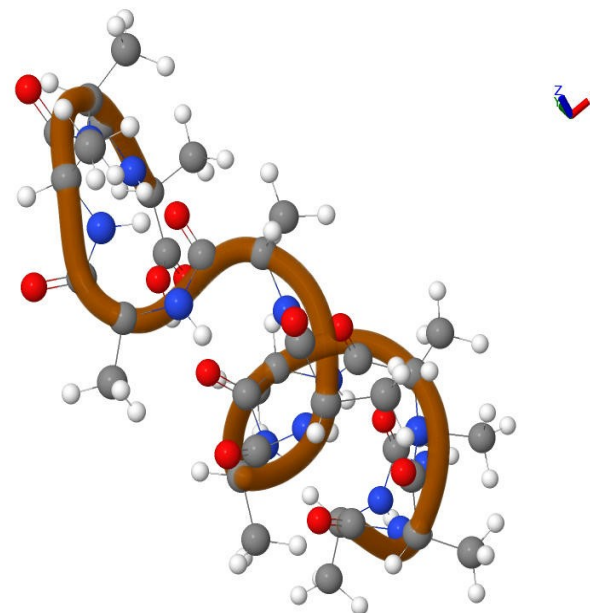
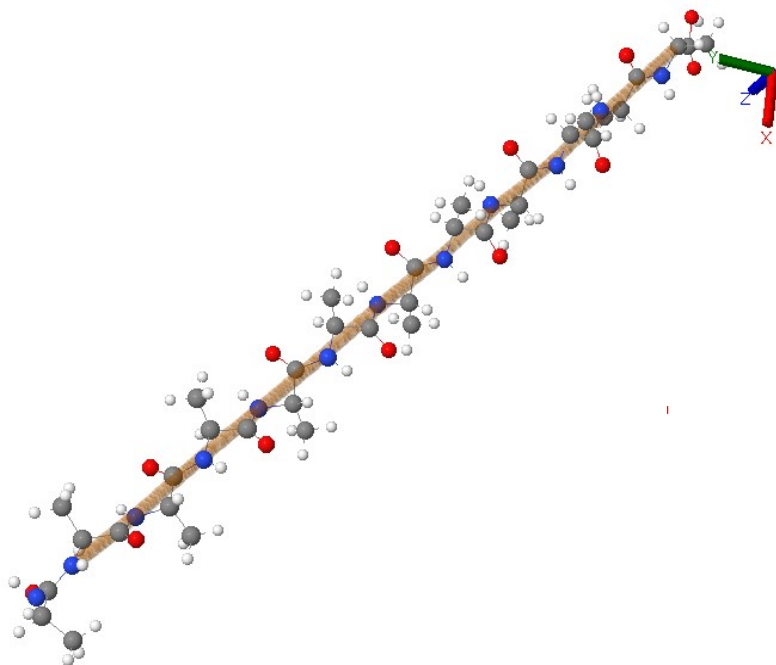
<http://de.wikipedia.org/wiki/Brennraum>  
 Urheber: „Softeis“, Lizenz „cc-by-sa“

<http://de.wikipedia.org/wiki/Sturm> (Public domain)



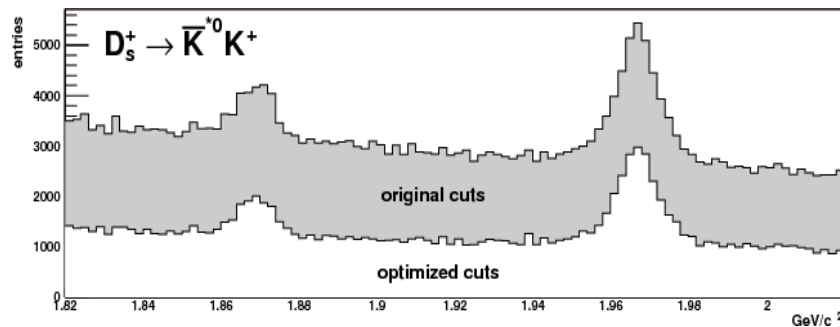
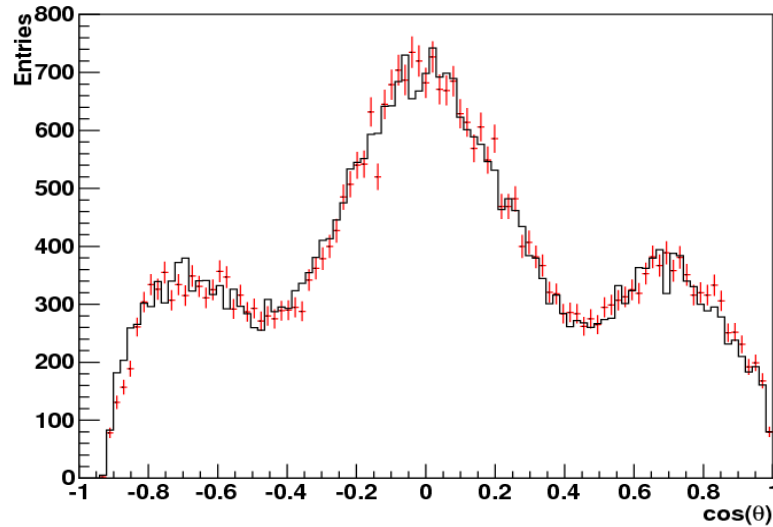


# Protein Folding

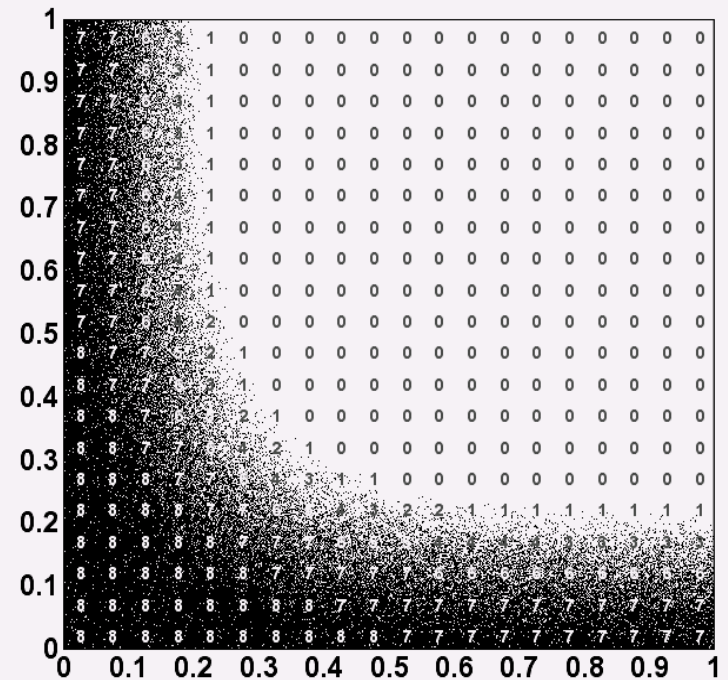


# Elementary Particle Physics, Neural Networks

cos( $\Theta$ ) @ heli data



Input and output of feedforward neural network (2-2-1)



Many problems can be described in terms of a set of parameters (e.g. floating point, Integer, boolean) and a *computer-implemented* evaluation function that assigns a (usually numeric) quality to them.

$$(x_1, x_2, \dots, x_n) \rightarrow f(x_1, x_2, \dots, x_n)$$

# Defining „parametric optimization“

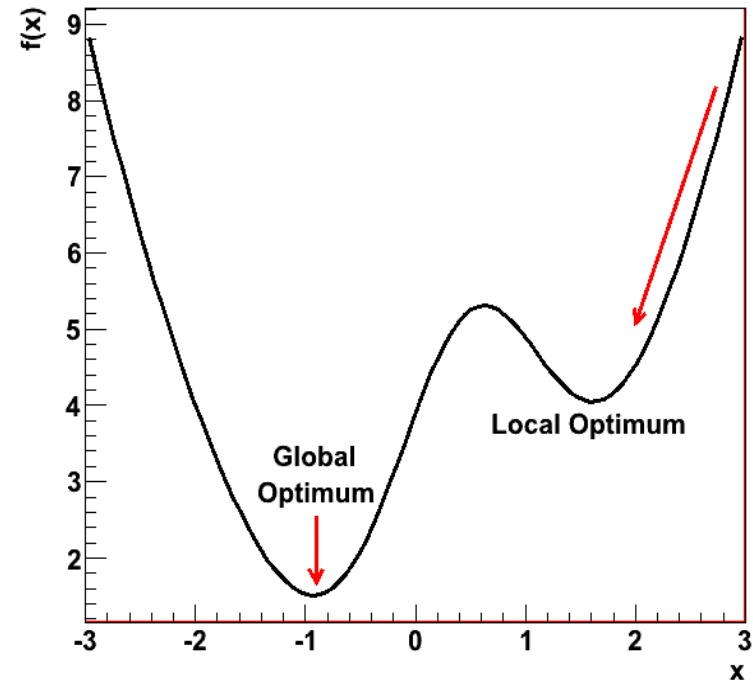
- In the context of this talk:

Parametric optimization refers to the selection of the best *available* element from a set of alternatives, according to a computer-implemented evaluation criterion  $f$

- Closely related to mathematical optimization:
  - Searching for extreme values (maxima or minima) of a function
  - But: We can usually not apply unmodified mathematical algorithms to  $f$
- In comparison: The *ideal*“ solution refers to the best possible result

## Some similarities

- There can be any number of local optima
- There can be many global optima (although more often there is just one)
- Some „traditional“ algorithms for searching minima/maxima of mathematical functions can be adapted to fit parametric optimization







Special algorithms are needed that avoid visiting every single parameter set (which would be impossible, as we have seen)

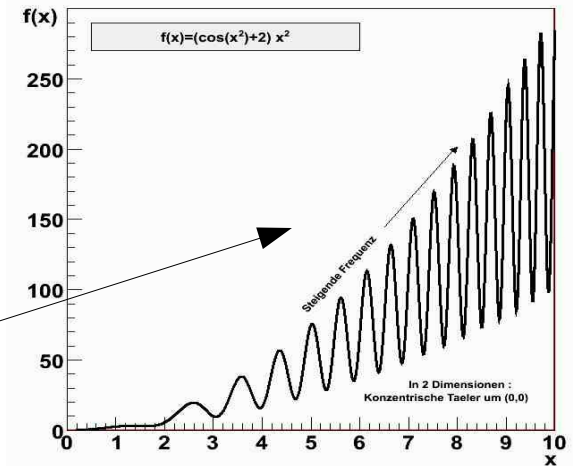
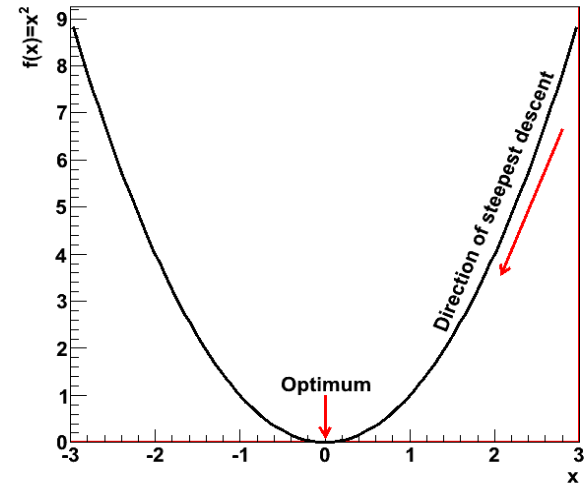
Even better, if they can be easily parallelized

...

# A simple solution

- Calculate the direction of steepest descent
  - Simple idea: „walk down-hill“
  - Can be done for computer-implemented evaluation criteria with an approximation

$$\frac{\partial f}{\partial x} \rightarrow \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$



This will fail!

# Evolutionary Strategies („ES“)

## ■ Algorithm:

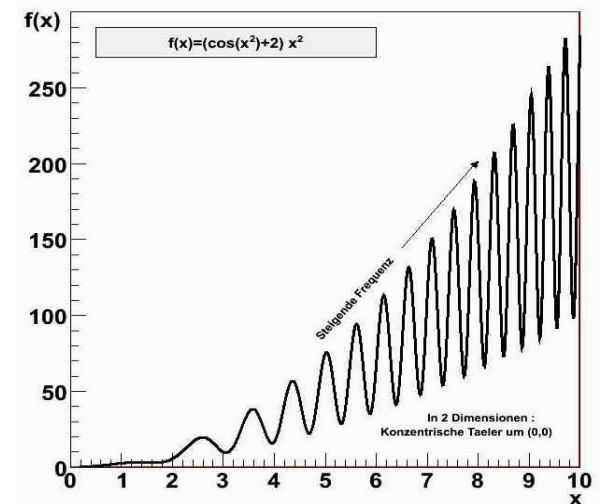
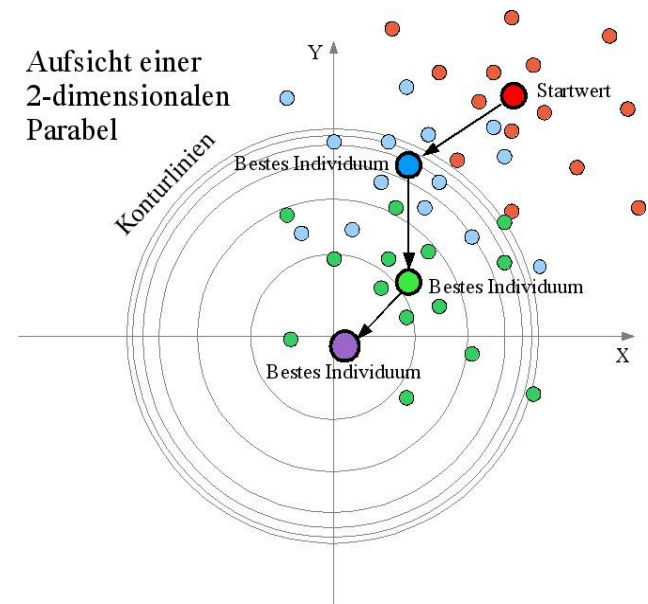
- Population of parents (best known solutions) and children
- Cycle of duplication, mutation, selection
- Mutation usually through addition of gaussian-distributed random numbers

## ■ Advantages:

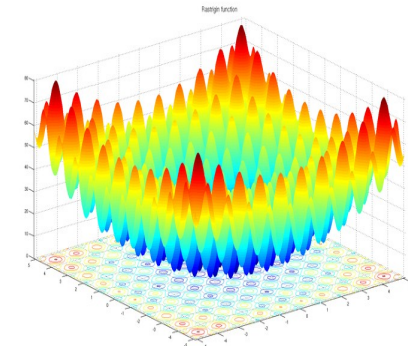
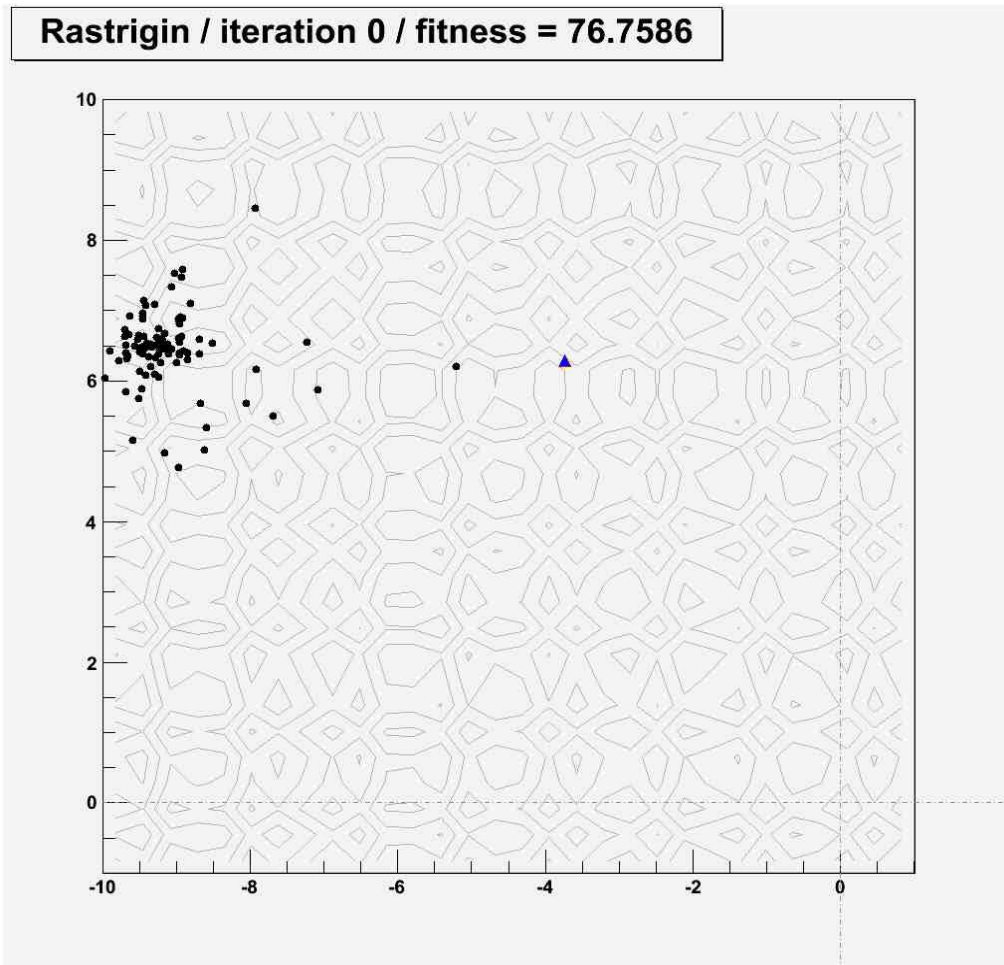
- Tolerant wrt. local optima
- Compute time scales with size of the population
- **Easy to parallelise**

## ■ Disadvantages

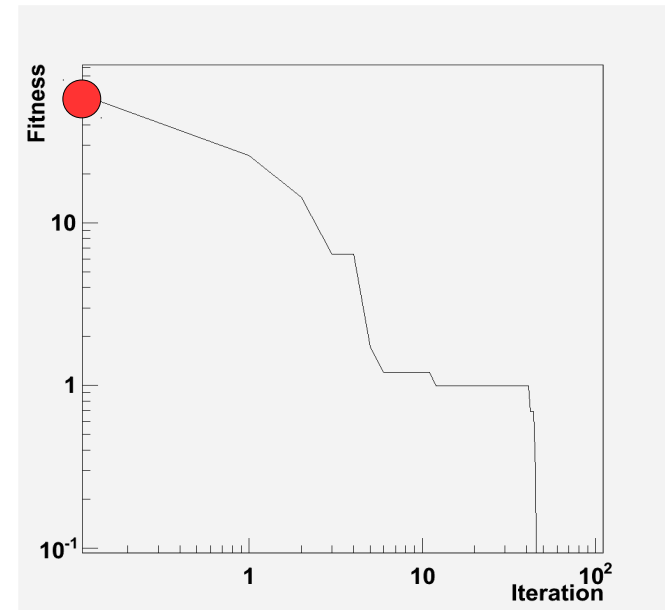
- Can be slower than gradient descent for smaller problems
- Many configuration options (e.g. width of gaussian)



# ES: The Rastrigin Function



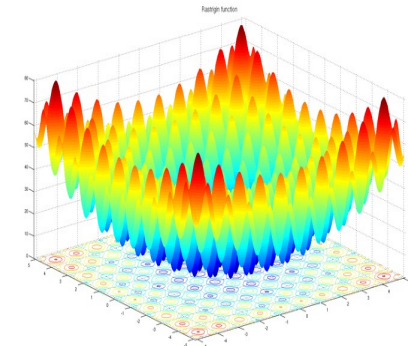
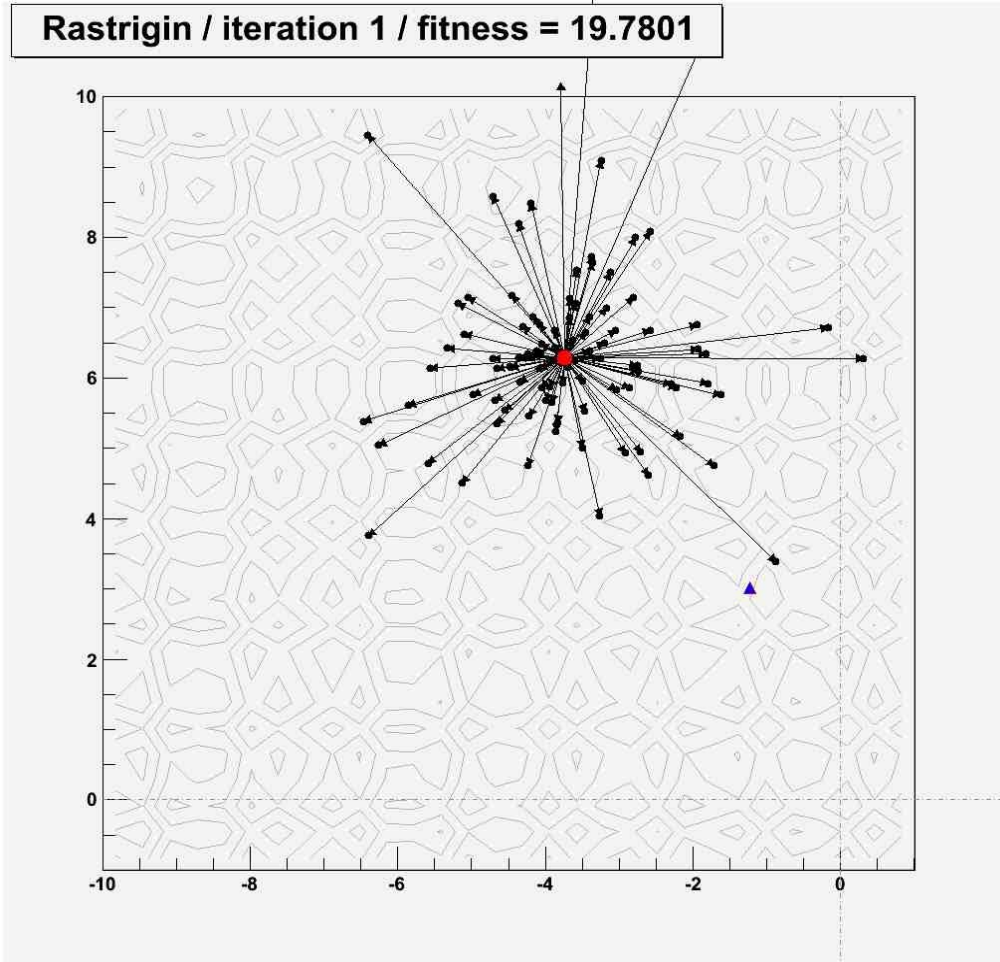
Picture: Wikipedia (public domain)



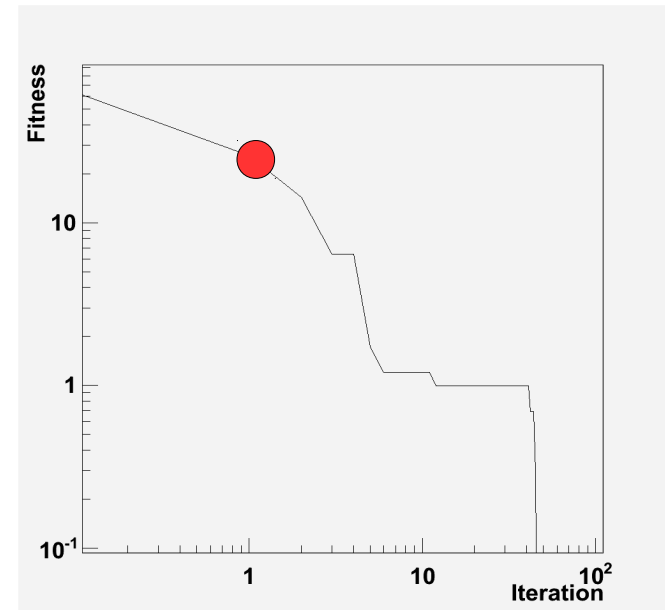
Done with Geneva; Plot created with the ROOT framework



# ES: The Rastrigin Function



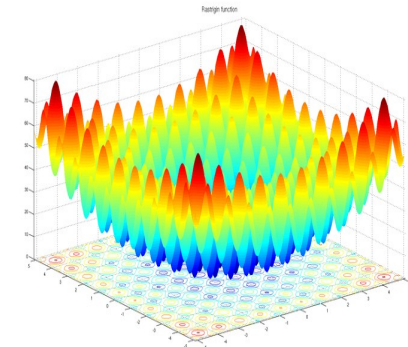
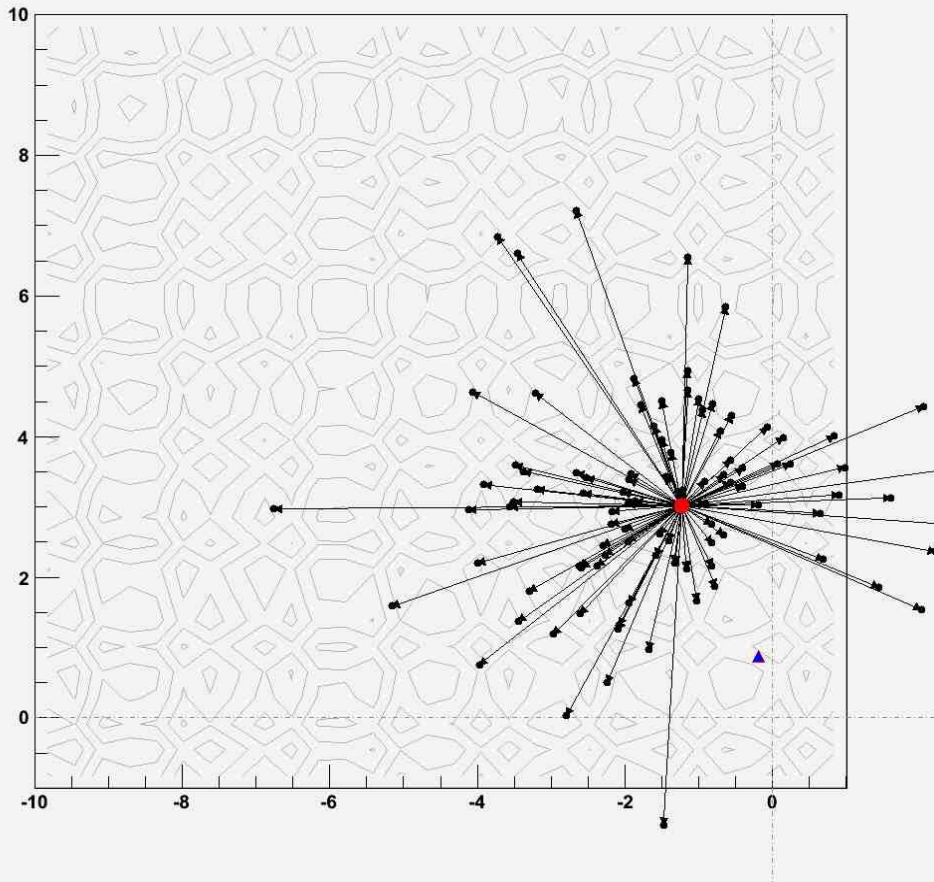
Picture: Wikipedia (public domain)



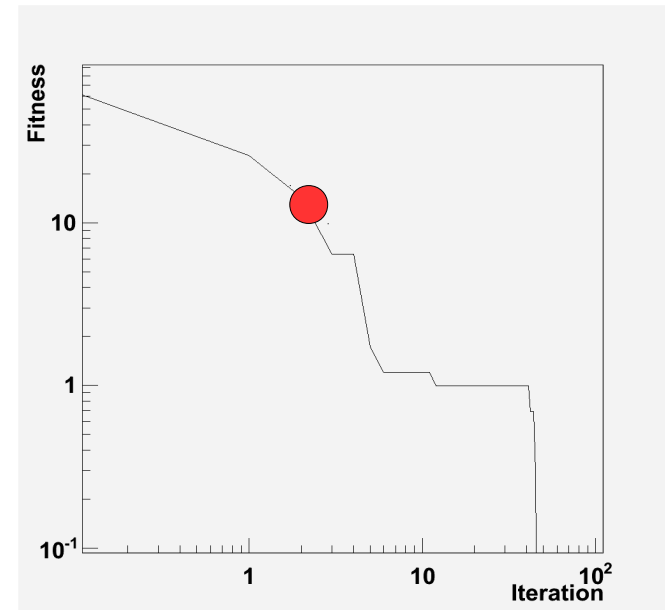
Done with Geneva; Plot created with the ROOT framework

# ES: The Rastrigin Function

Rastrigin / iteration 2 / fitness = 10.0394



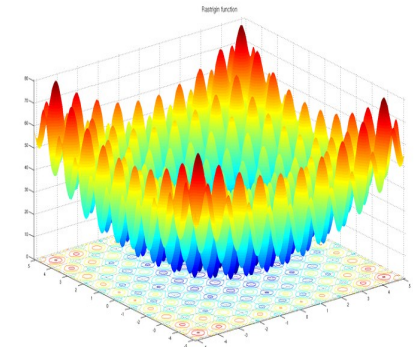
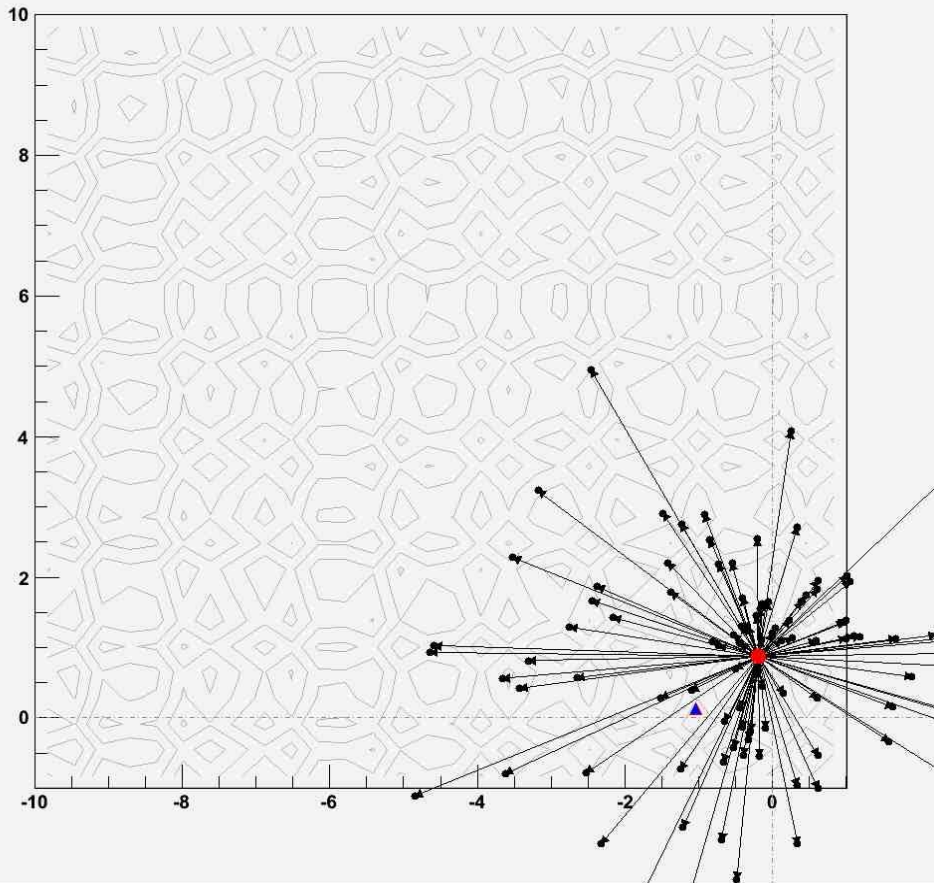
Picture: Wikipedia (public domain)



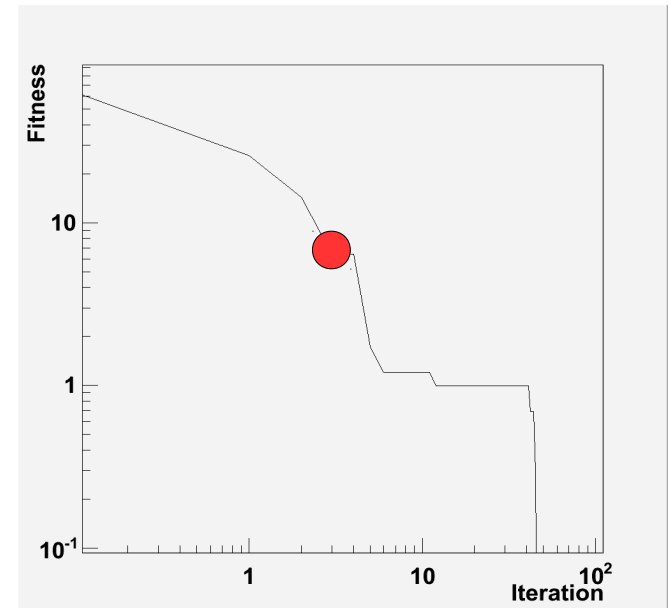
Done with Geneva; Plot created with the ROOT framework

# ES: The Rastrigin Function

Rastrigin / iteration 3 / fitness = 4.56426



Picture: Wikipedia (public domain)



Done with Geneva; Plot created with the ROOT framework

# Other Algorithms

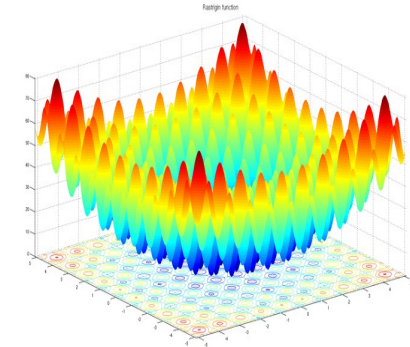
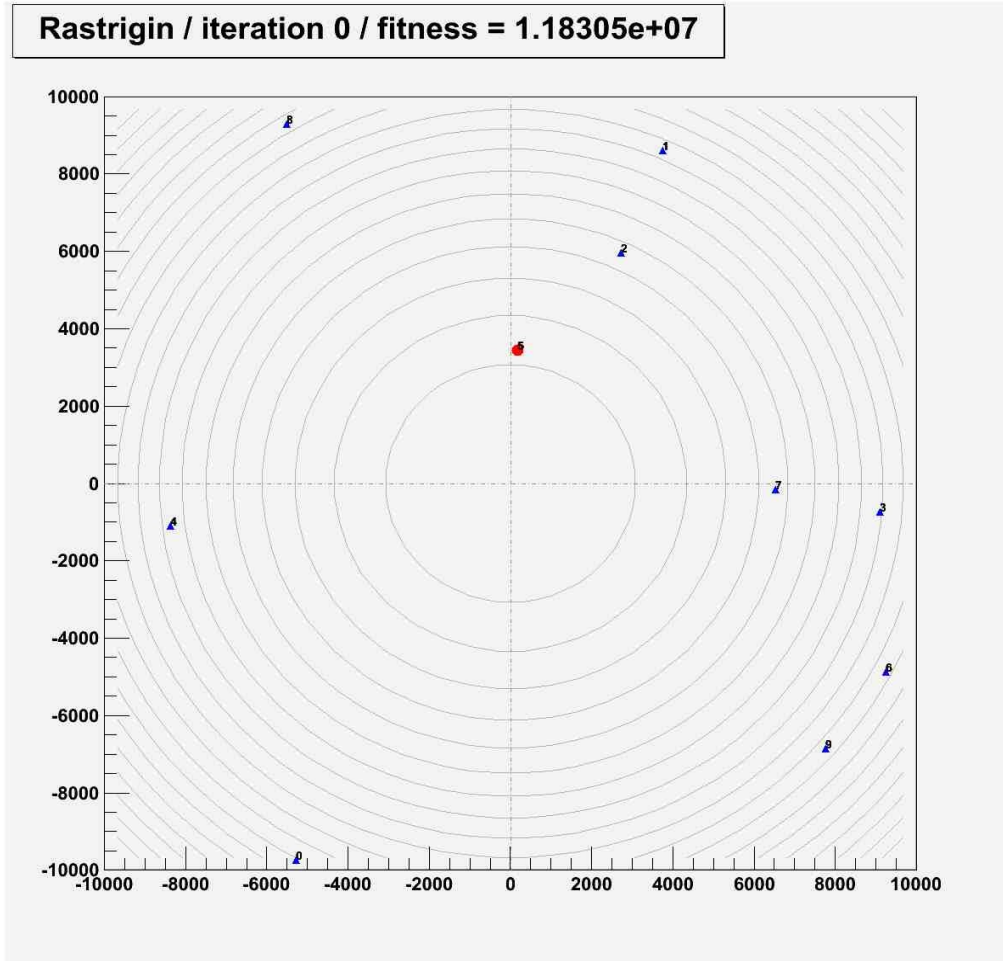
- Particle Swarm Algorithms
  - Members of „neighborhoods“ of candidate solutions are drawn in each iteration towards
    - The globally best solution
    - The best solution of the neighborhood
    - A random direction
- Deluge algorithms / Simulated Annealing
- Line search, Simplex, ...



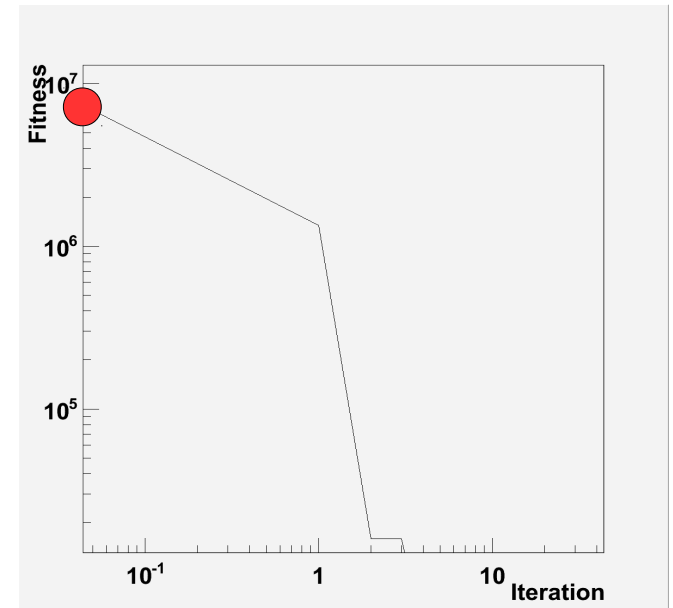
(Source: Wikipedia; Author D. Dibenski; public domain)



# Swarm: Rastrigin Function



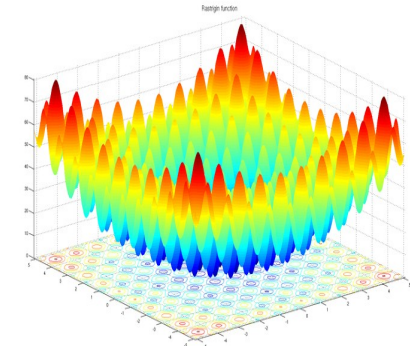
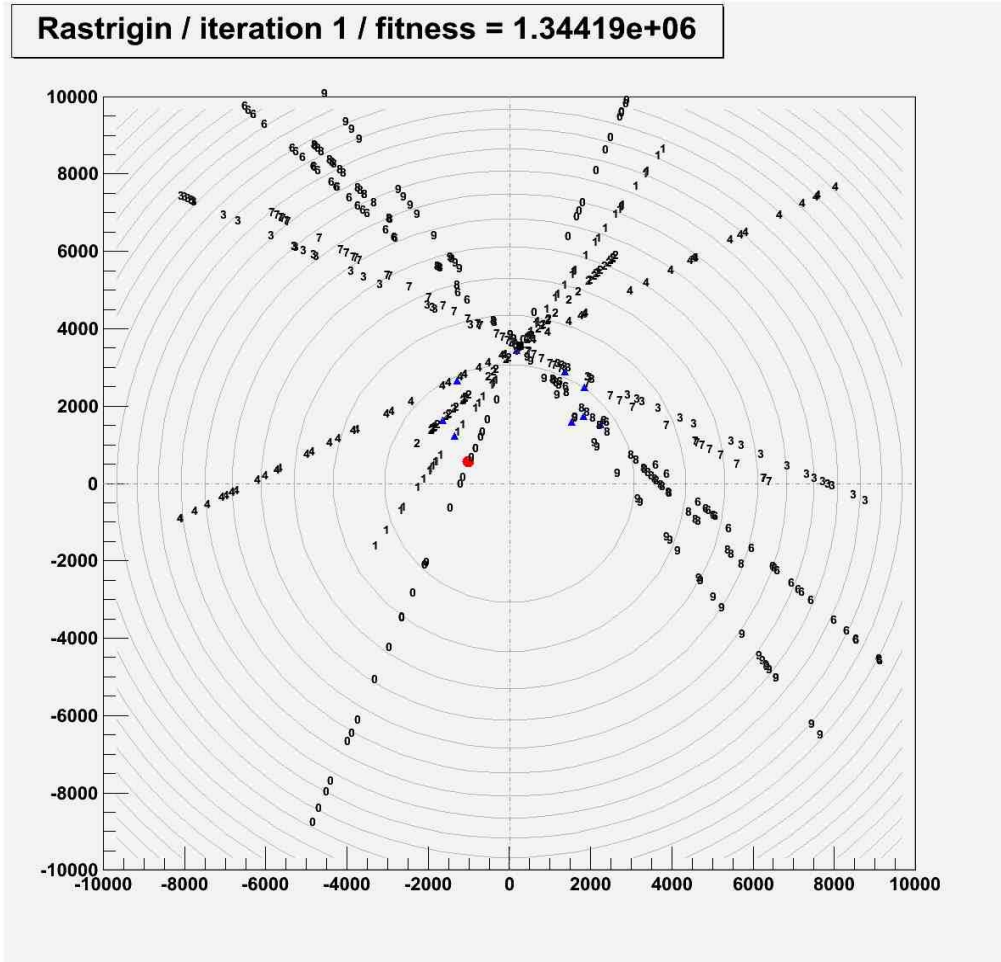
Picture: Wikipedia (public domain)



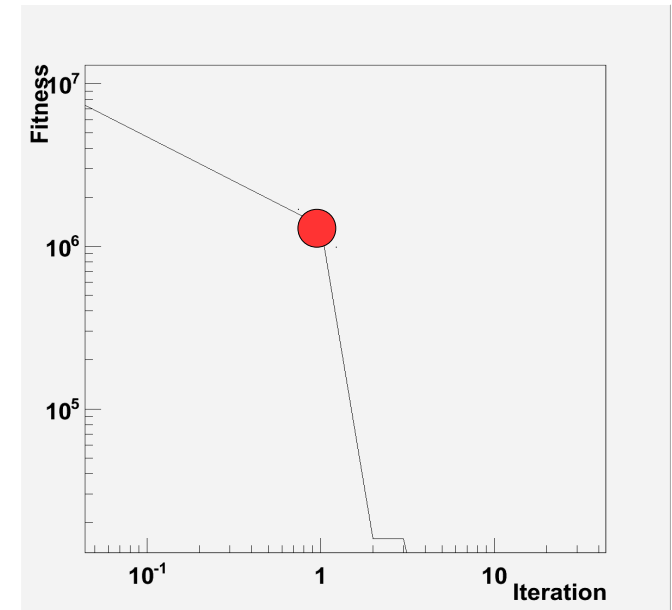
Done with Geneva; Plot created with the ROOT framework



# Swarm: Rastrigin Function

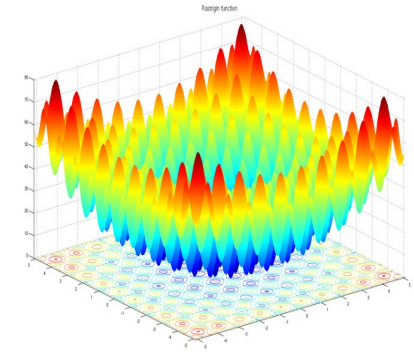
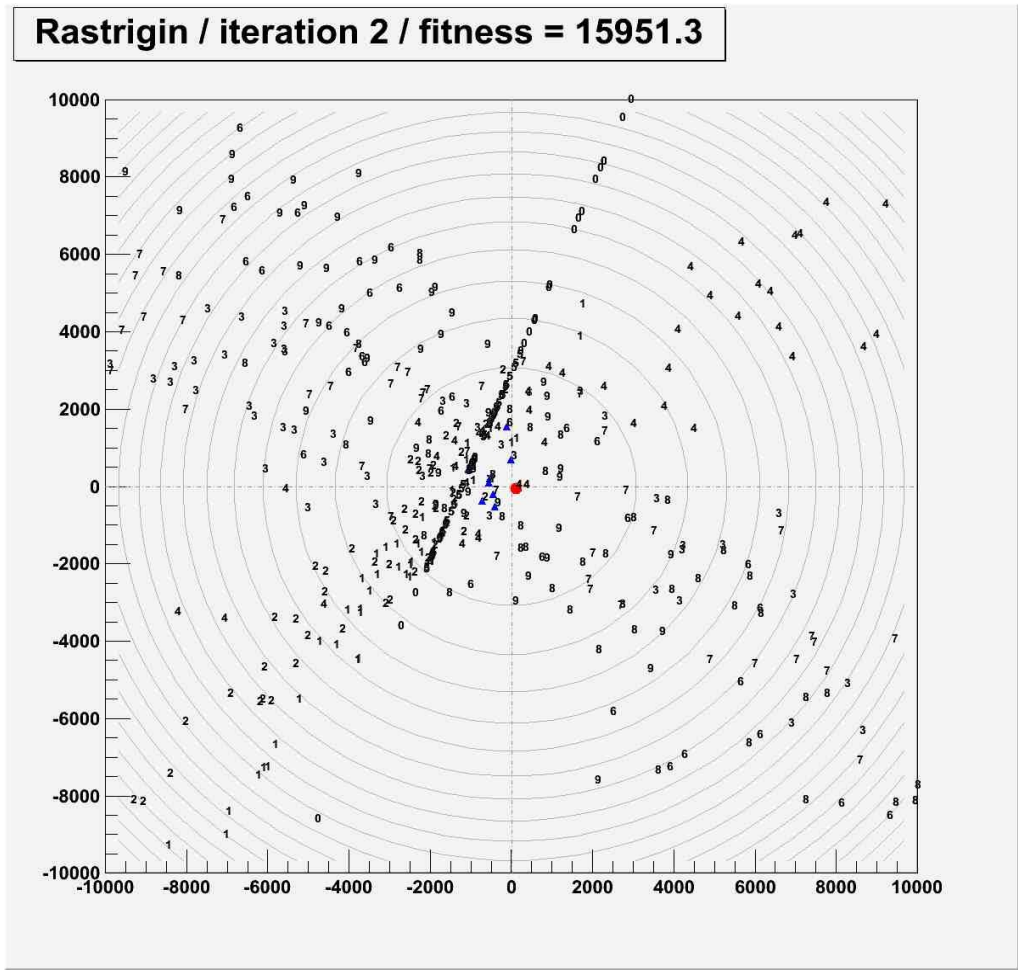


Picture: Wikipedia (public domain)

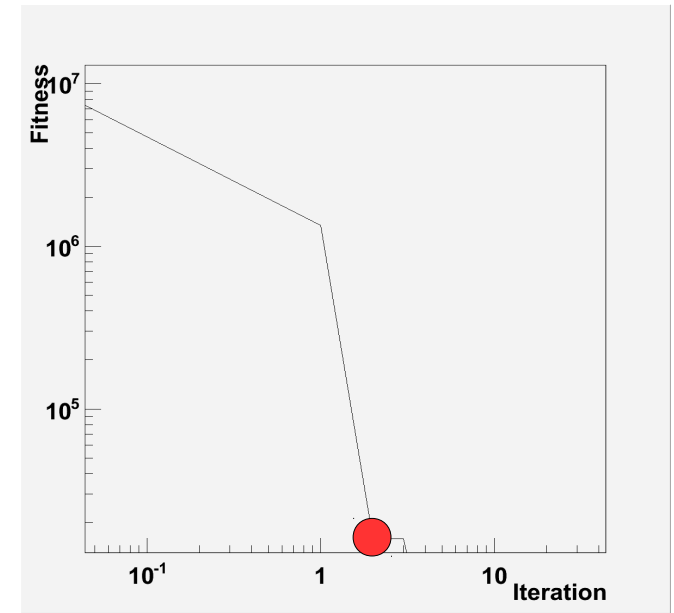


Done with Geneva; Plot created with the ROOT framework

# Swarm: Rastrigin Function

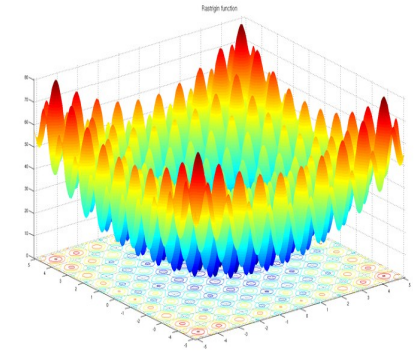
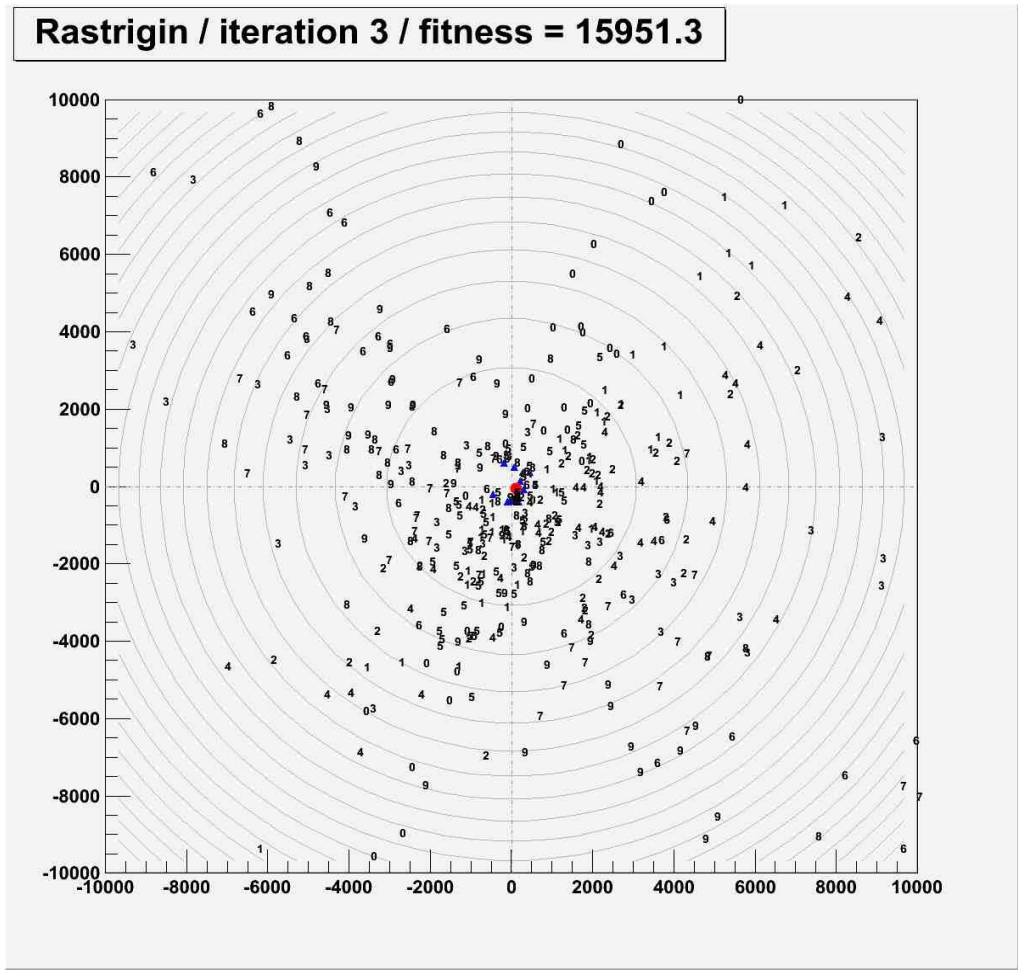


Picture: Wikipedia (public domain)

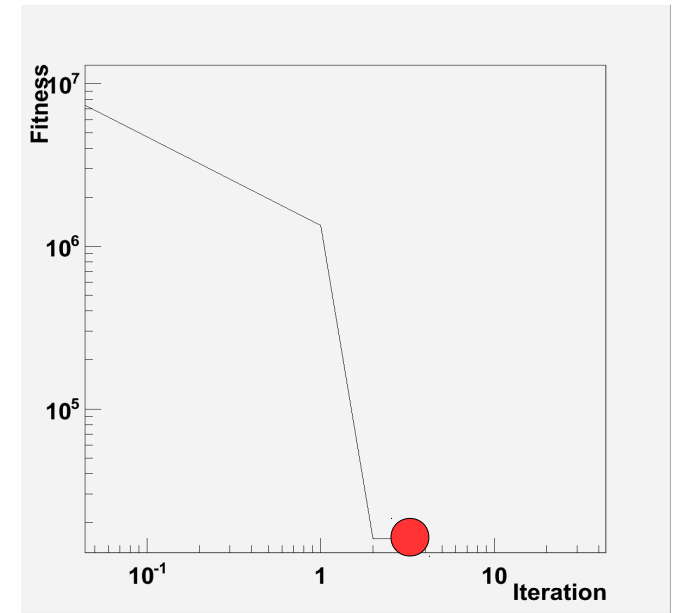


Done with Geneva; Plot created with the ROOT framework

# Swarm: Rastrigin Function



Picture: Wikipedia (public domain)



Done with Geneva; Plot created with the ROOT framework

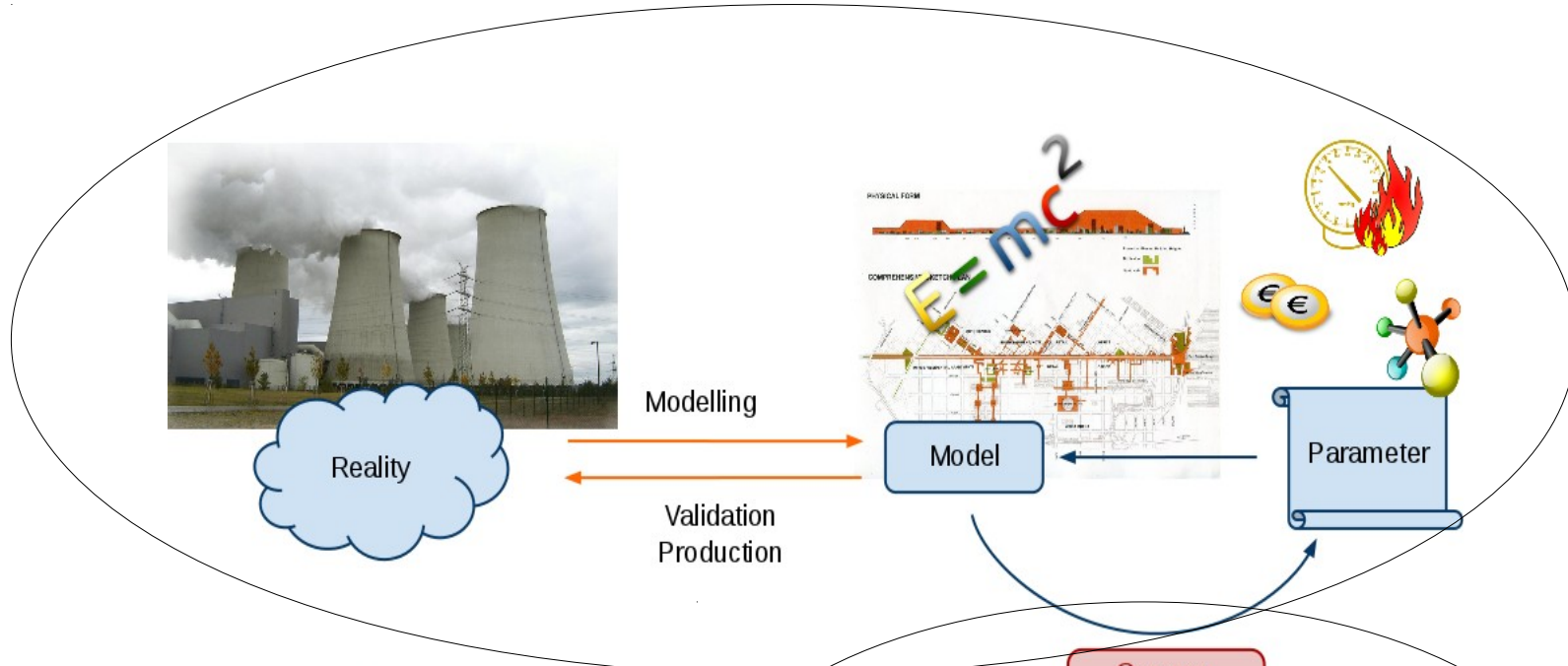
The examples above were calculated  
with the Geneva library  
of optimization algorithms

# Geneva: Boundary Conditions

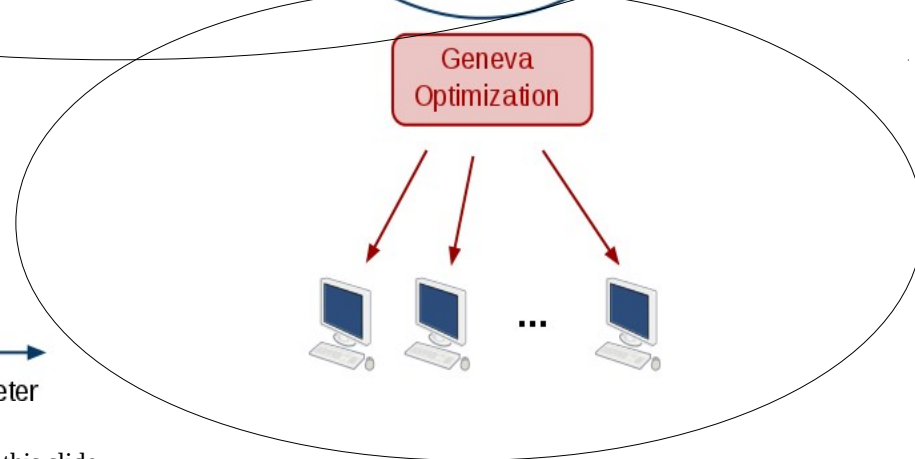
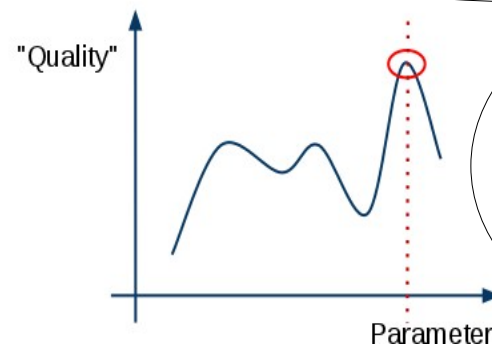
- Geneva wants to provide users with an environment that lets them solve optimization problems of any size transparently, as easily on a single core-machine as in the Grid or Cloud.
- Geneva wants to become a „warehouse of optimization algorithms“, all working on the same problem descriptions
- Geneva targets optimization problems, whose figure of merit requires long-lasting computations
- We assume that many very large scale optimization problems so far have not been targetted as
  - Typical single- or multi-core machines do not offer sufficient computing power
  - The complexities of running optimizations in parallel and/or distributed enviroments lead to the assumption that performing such computations is not feasible



Problem definition



Optimization



# Geneva: Design Criteria

- Focus on long-lasting, computationally expensive evaluation functions
  - *Stability of core library* rated higher than efficiency
  - Suitable for distributed environments
- Serial, multi-threaded and networked execution, transparent to users
  - Implications of networked and multi-threaded execution:
    - No global variables
    - User-defined data structures must be serializable
- Familiar interface
  - STL interface for data, individuals, populations, ...

# Geneva: Design Criteria

- Fault tolerance of networked execution:
  - Algorithm must be able to repair itself in case of missing or late replies from clients
- Execution of clients in Grid and Cloud:
  - No push mode means: Server needs public IP, clients don't
- Easy, portable build environment:
  - CMake
- Quality assurance:
  - Unit-tests, based on Boost.Test library
  - Can be integrated into user code

# Geneva: Implementation

- C++
  - Heavily uses Boost
  - Efficient (cmp. Java)
  - About 80000 LOC
- So far largely Linux-based
  - But likely portable
  - Tested with various g++, Intel
- Major components
  - Representation of parameter sets
  - Optimization algorithms
  - Parallelization and communication
  - Random number factory

```
int main(int argc, char **argv)
{
    GOptimizer go(argc, argv);

    //-----
    // Client mode
    if(go.clientRun()) return 0;

    //-----
    // Server mode

    // Create the first set of individuals.
    for(std::size_t p = 0 ; p<nParents; p++) {
        boost::shared_ptr<GParameterSet>
            functionIndividual_ptr
                =
                GFunctionIndividual<>::getFunctionIndividual();

        // Make the param. collection known to individual
        go.push_back(functionIndividual_ptr);
    }

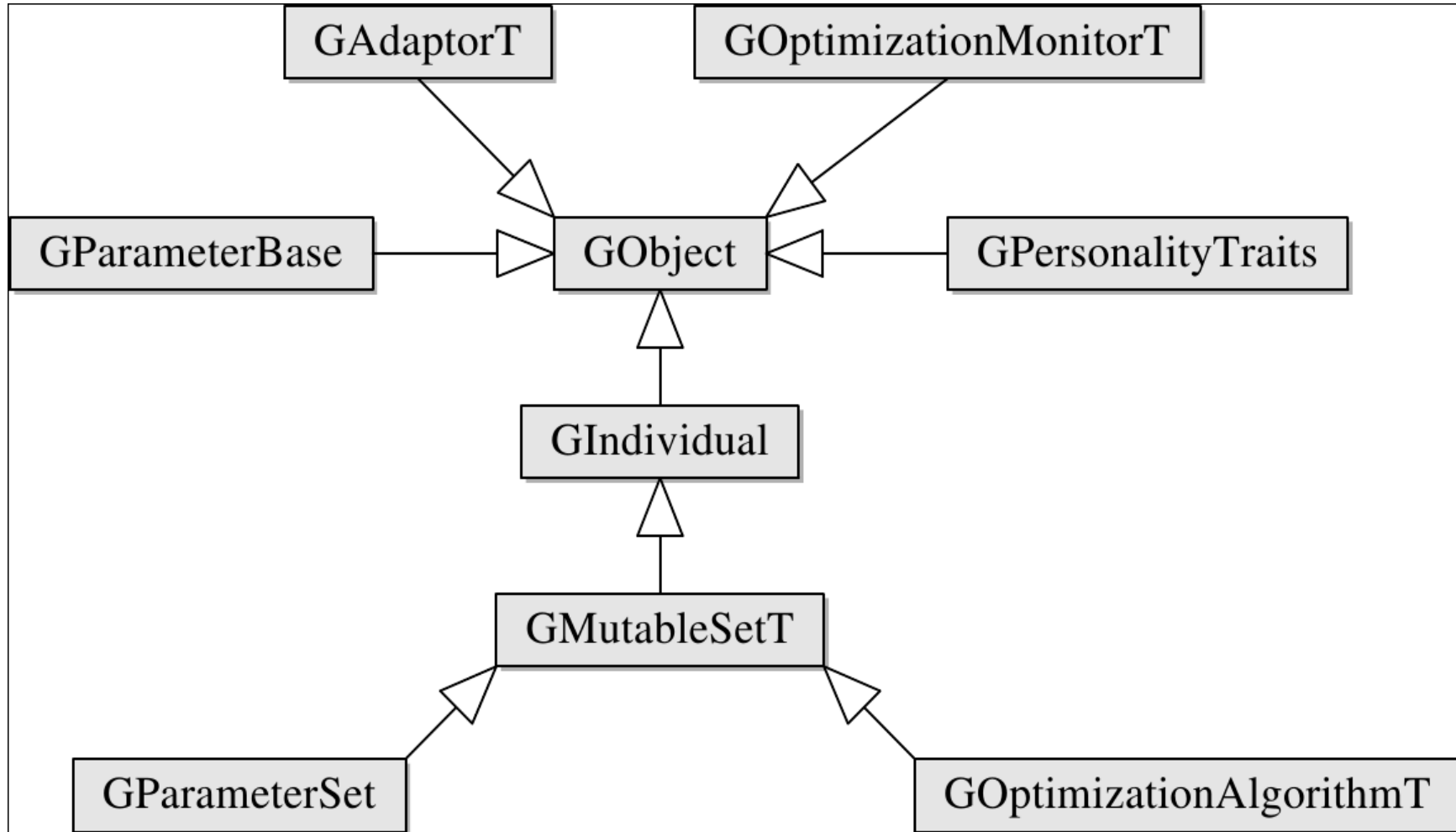
    // Perform the actual optimization
    boost::shared_ptr<GParameterSet>
        bestFunctionIndividual_ptr
            = go.optimize();

    // Do something with the best individual
    // [...]

    std::cout << "Done ..." << std::endl;
    return 0;
}
```

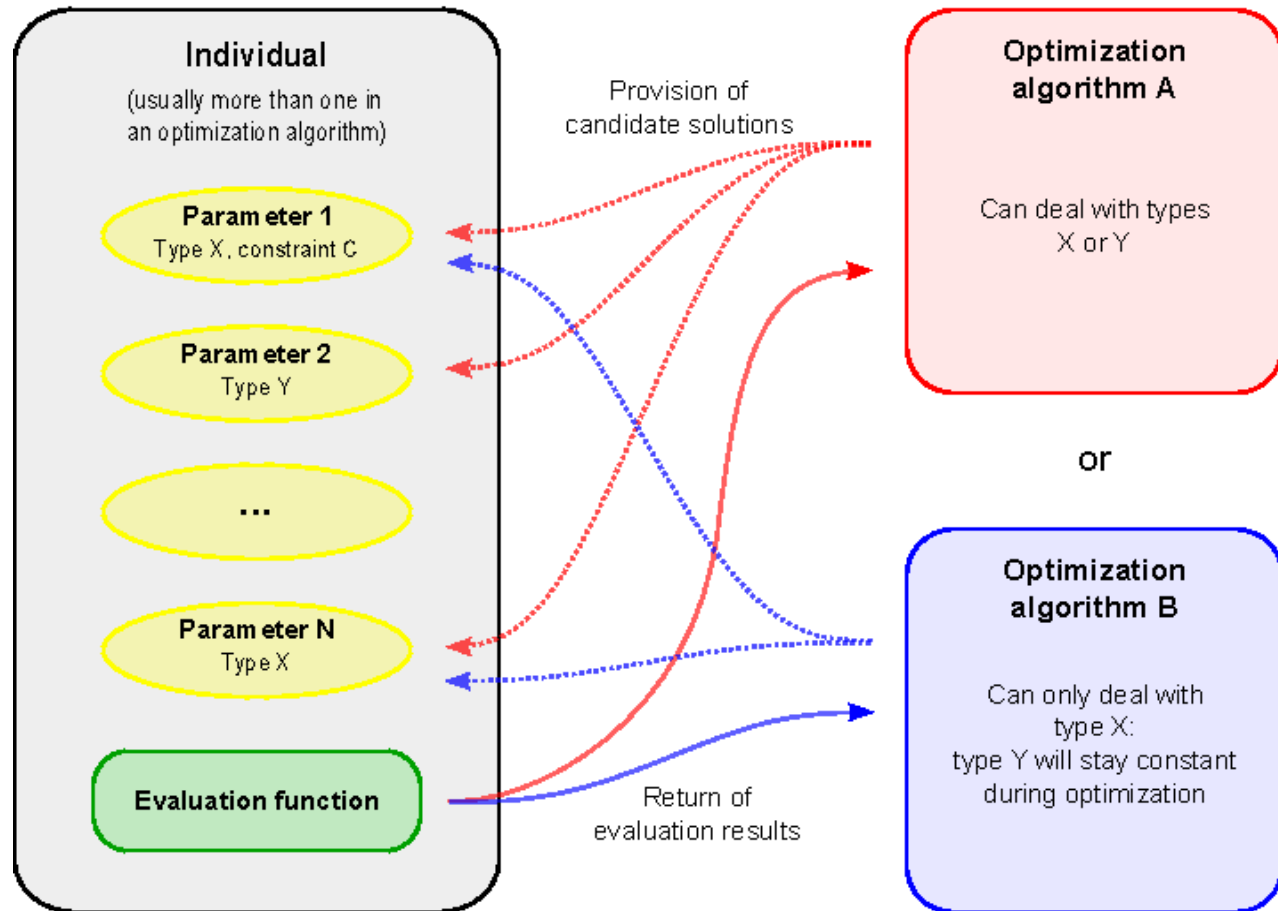




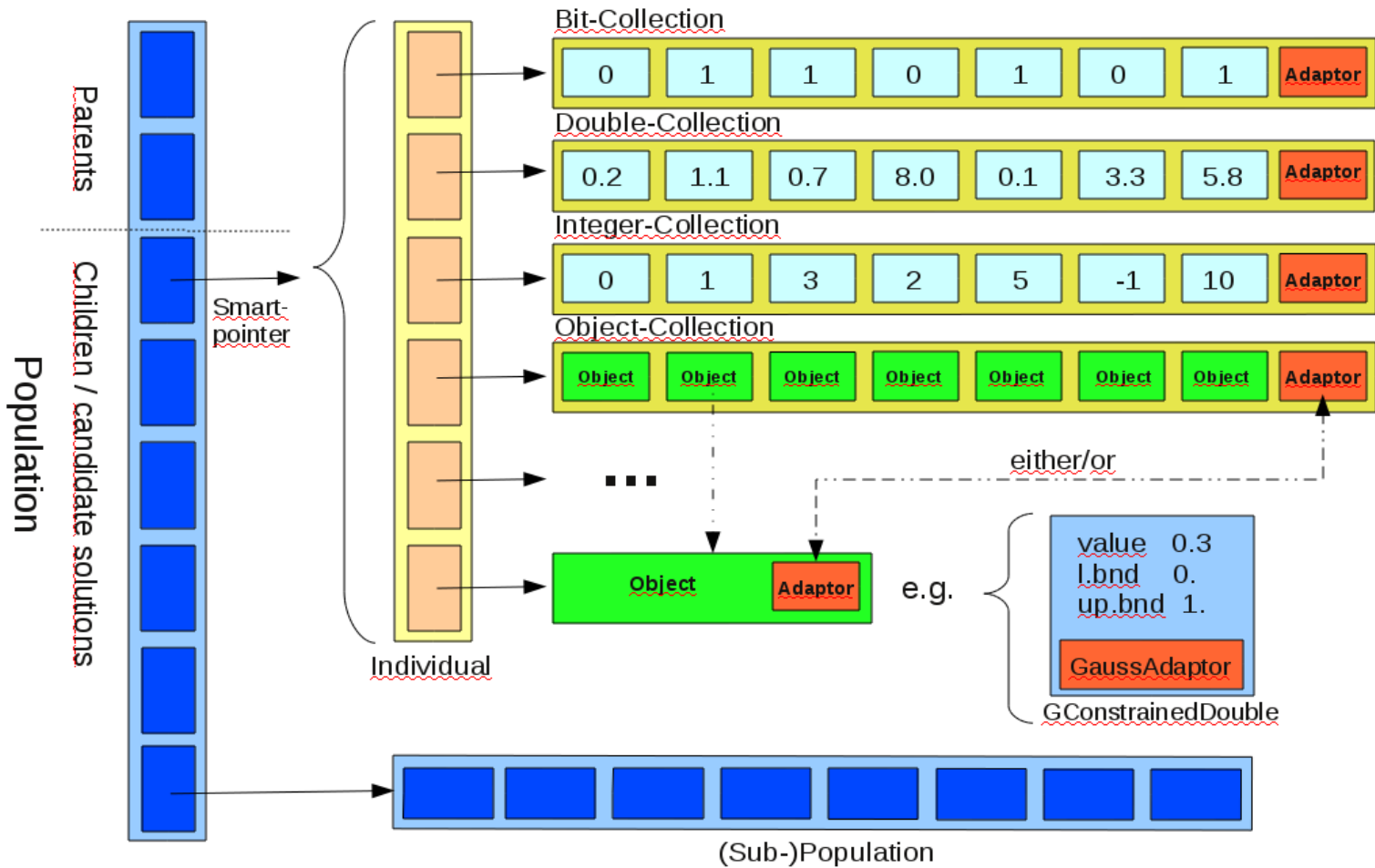


# Parameters vs. Algorithm

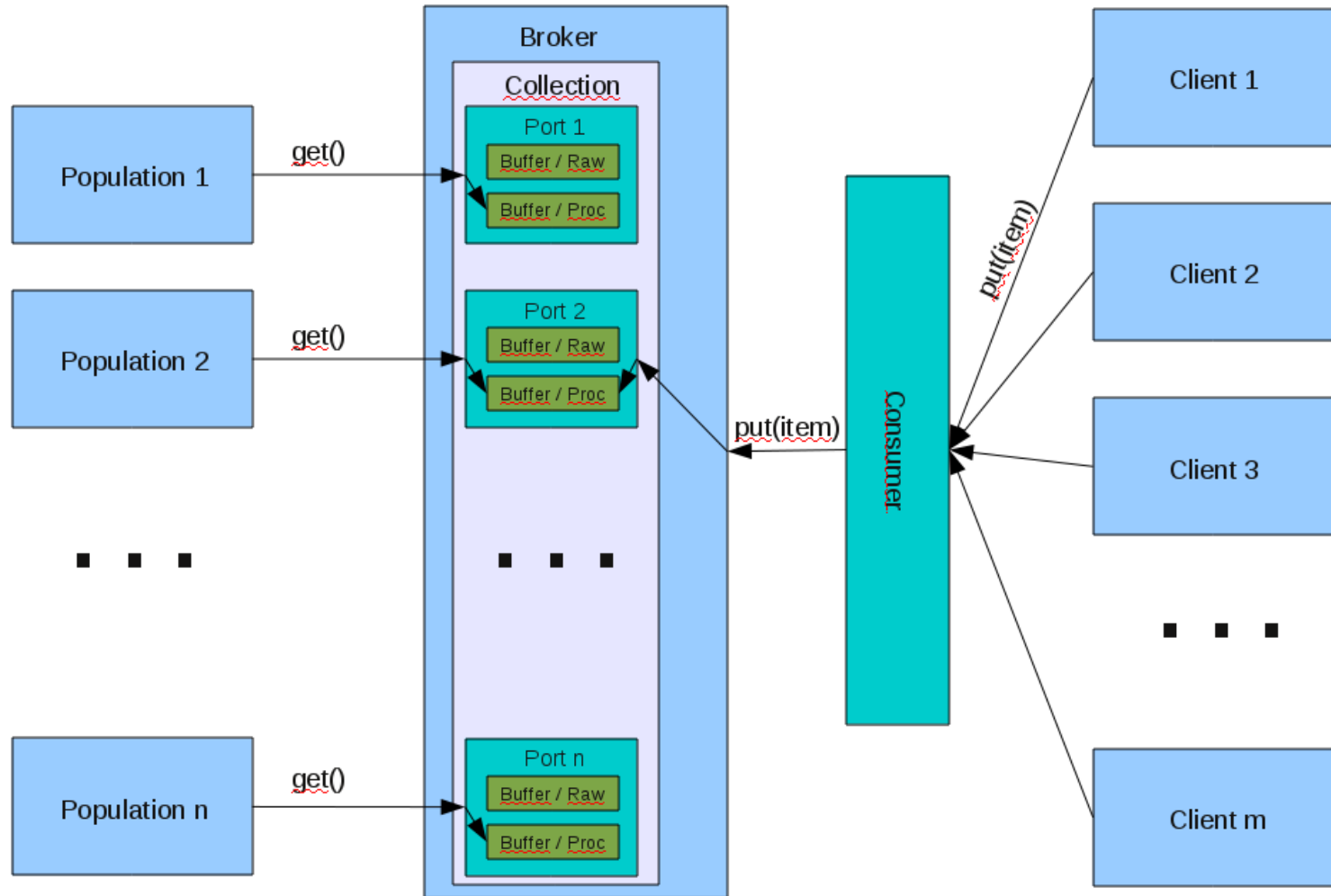
## Iteration-based optimization



# ES: Data Representation / Mixed Parameter Sets



# The Broker



# Capabilities

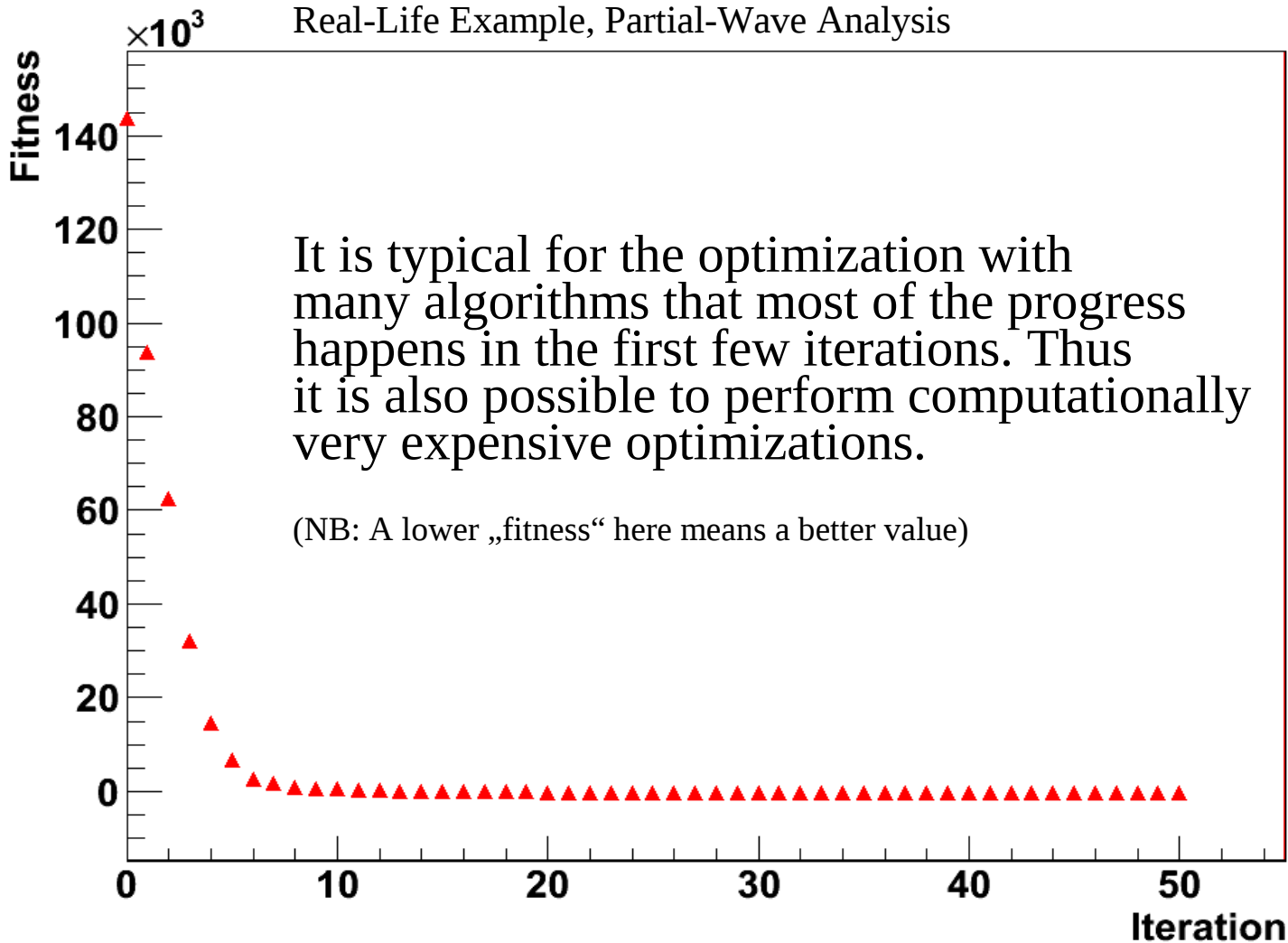
- Geneva today implements
  - Evolutionary Algorithms
  - Particle Swarm Optimization
  - Gradient Descent
  - Simulated Annealing
- Execution in
  - Serial mode
  - Multithreaded mode
  - Networked mode
- Mixed problem descriptions
  - May contain different parameter types
- Meta-Evolution



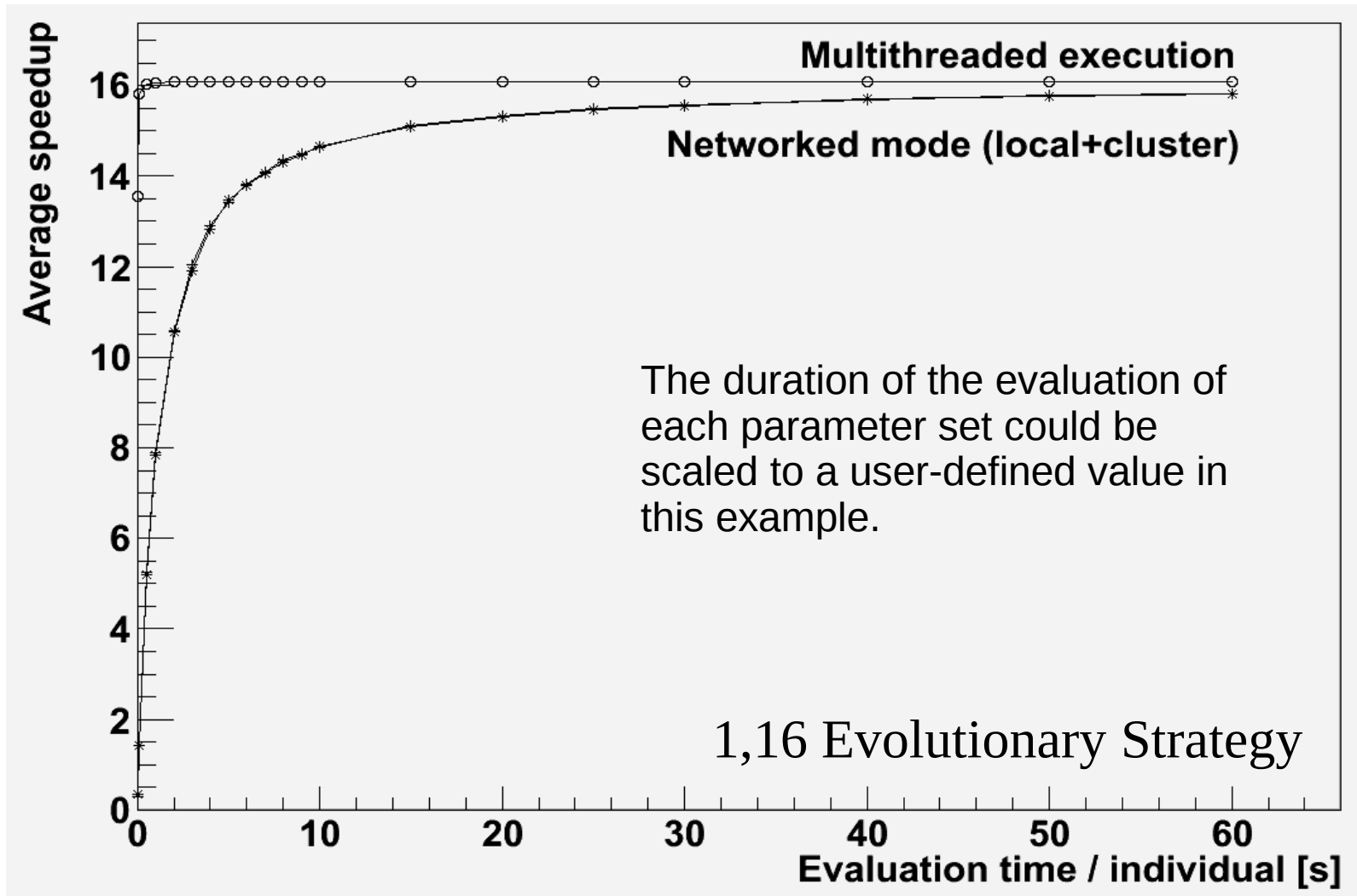
```
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
top - 16:03:19 up 118 days, 5:21, 2 users, load average: 44.03, 30.19, 14.96
Tasks: 577 total, 1 running, 576 sleeping, 0 stopped, 0 zombie
Cpu(s): 99.9%us, 0.1%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 98998852k total, 13273640k used, 85725212k free, 366152k buffers
Swap: 101056504k total, 40224k used, 101016280k free, 11412904k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
25879 berlich 17 0 2684m 668m 28m S 4803.0 0.7 97:24.42 GOmegaPiApp
25868 berlich 15 0 13140 1492 820 R 5.8 0.0 0:02.31 top
25603 berlich 15 0 90128 1824 1088 S 0.0 0.0 0:00.08 sshd
25604 berlich 15 0 66060 1620 1252 S 0.0 0.0 0:00.05 bash
25653 berlich 15 0 90128 1828 1088 S 0.0 0.0 0:00.06 sshd
25654 berlich 15 0 66060 1596 1240 S 0.0 0.0 0:00.01 bash
```

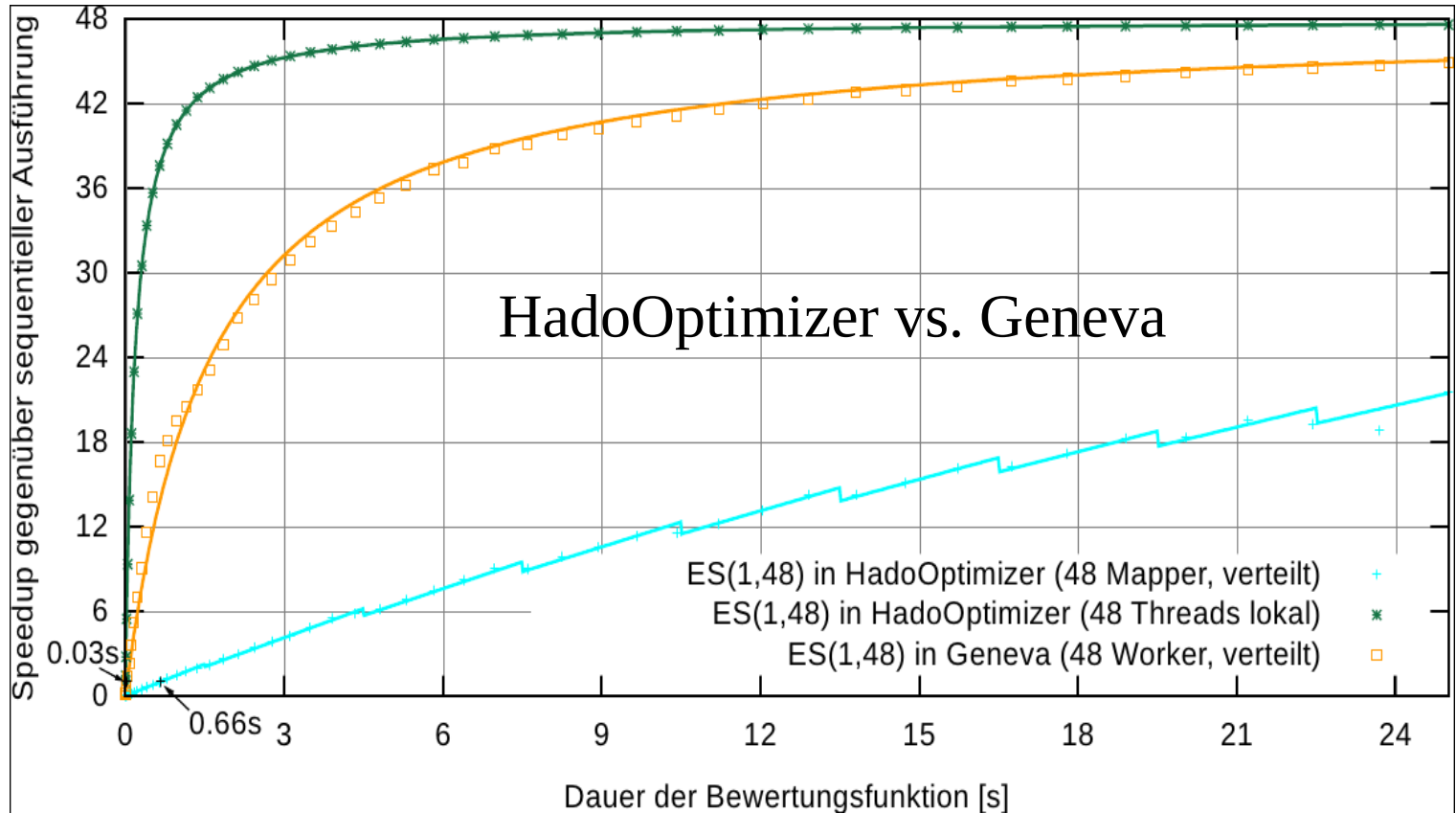
- 48 Core AMD Sytem (Magni Cours)
- Here: Evolutionary Algorithms
- Example from High-Energy Physics (Partial Wave Analysis / RUB, Uni Mainz, GSI Darmstadt)



# Performance: Speedup in a Cluster

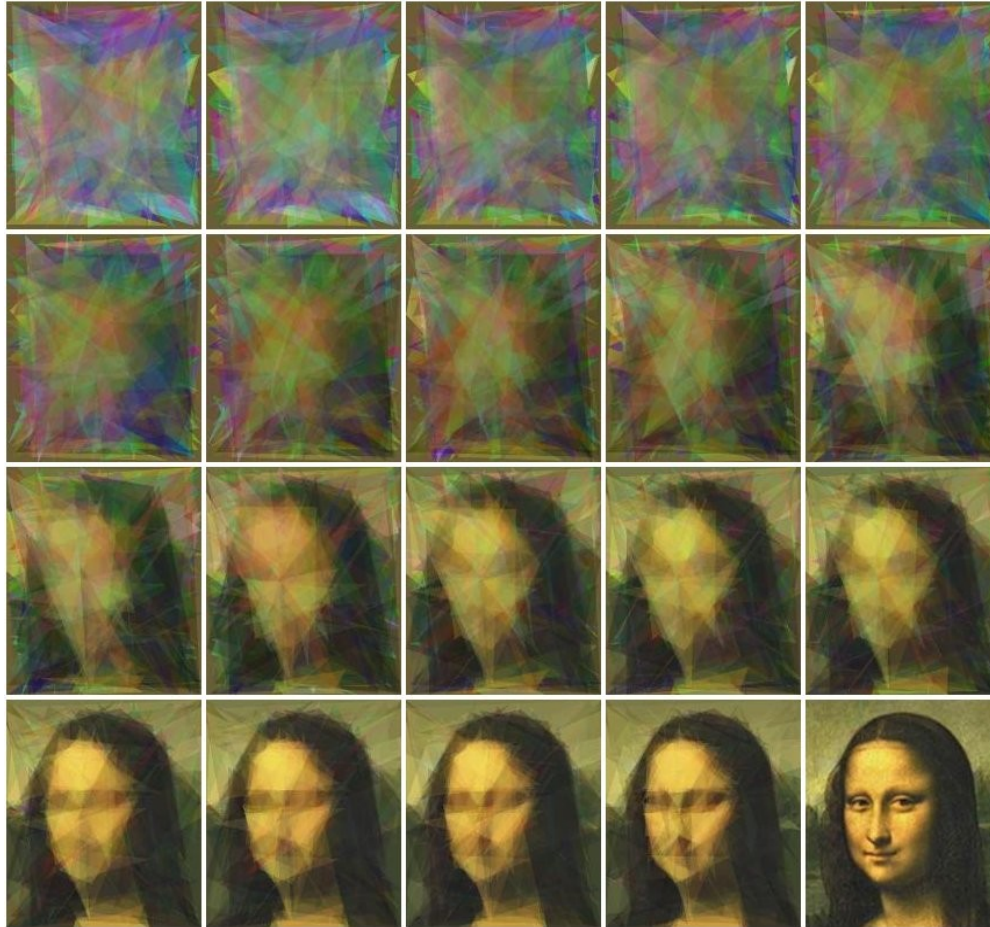


# Performance: Comparison with Hadoop ES implementation (Java)



Thanks to Christian Kumpe for the permission to use this plot

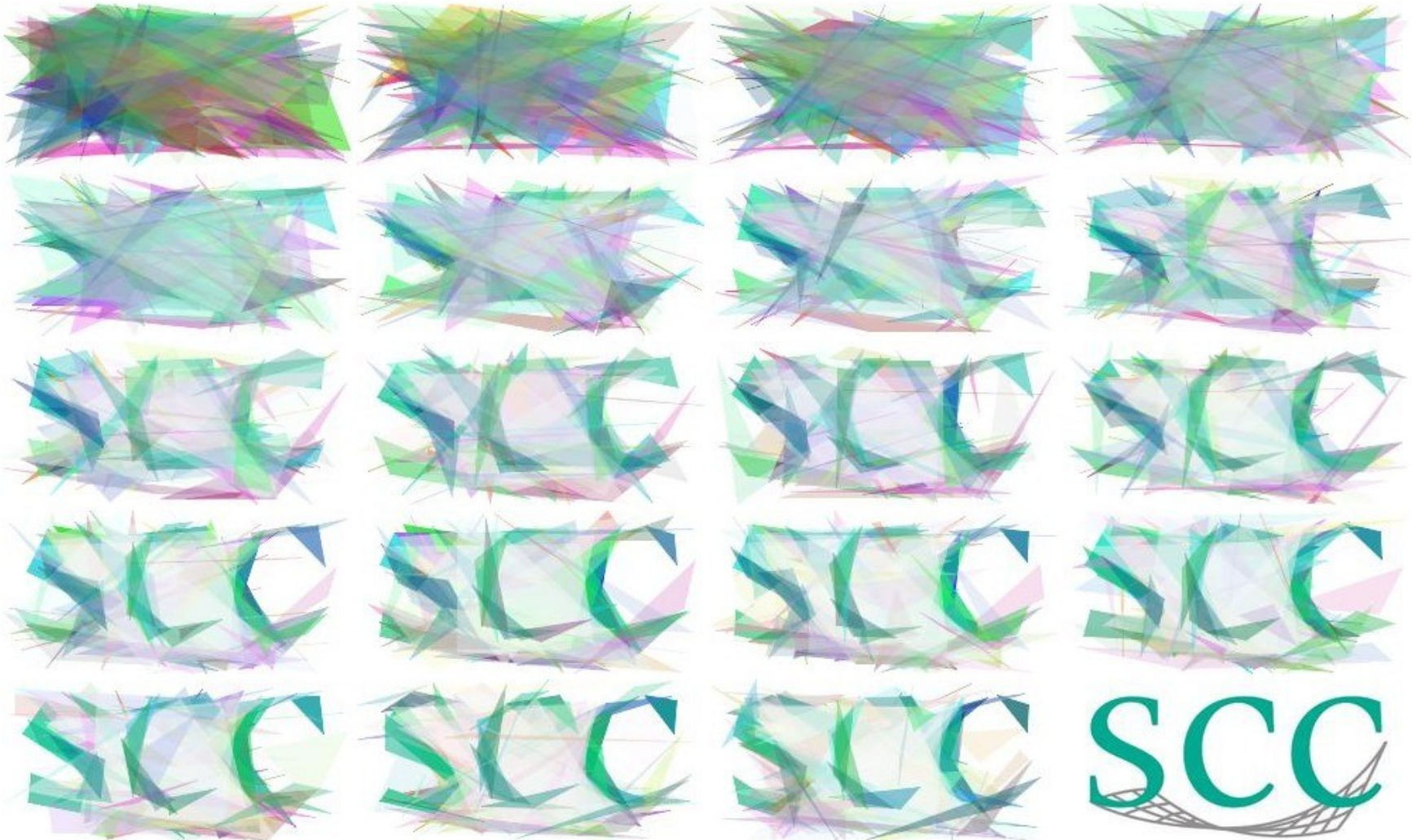
# Modelling the Mona Lisa



- Goal of the optimization
  - 300 semi-transparent triangles should be superimposed in such a way that they resemble a given target picture
  - 10 parameters / triangle: alpha-channel, coordinates and colors
  - Means that suitable values for 3000 parameters must be found, with no known start value



# Modelling the SCC Logo



# Geneva and Boost

- Only real external dependency
- Geneva uses (in no particular order)
  - ASIO
  - Serialization
  - Threads
  - Smart pointers
  - Casts
  - Bind, Function, ...
- Has tremendously sped up development
  - First approach in case of some required feature:  
Check the Boost libraries, whether it already exists
- **Development of Geneva would not have been possible without Boost**

# Geneva is Open Source

- No vendor lock-in
- Availability of source code independent of success of Gemfony
- Full control over what happens
- Customer-driven development
  - Can benefit from third-party suggestions
  - Can initiate own ideas
- Users can sometimes be suspicious ...

- Shorter Development Cycles
  - Parallelization
  - Good results often within first few cycles
- Robust Optimization Algorithms
  - Local optima
  - Missing responses
- **ES probably the most mature algorithm!**
- Good scalability for computationally expensive problems
  - Ease of use of distributed resources
  - Better coverage of available resources
- Wide spectrum of deployment scenarios – virtually generic
  - Independence of optimization algorithm and optimization problem
- **None of this would have been possible without Boost**

Questions ?

# Questions!

[r.berlich@gemfony.com](mailto:r.berlich@gemfony.com)

# Thank you!

We would like to thank the organizers of this event, the audience, the sponsors of Geneva's development (particularly Karlsruhe Institute of Technology, Steinbuch Centre for Computing, and the Helmholtz Gemeinschaft Deutscher Forschungszentren)



- The slide „Connecting fields of knowledge“ contains several icons and pictures whose license should be named:
  - CC-BY:
    - <http://www.flickr.com/photos/matthias17/1725198264/>
    - <http://www.flickr.com/photos/walkingsf/4468474849/>
  - Public domain images from Clker.com:
    - <http://www.clker.com/clipart-mass-energy-equivalence-formula.html>
    - <http://www.clker.com/clipart-49393.html>
    - <http://www.clker.com/clipart-euro-coins.html>
    - <http://www.clker.com/clipart-molecule.html>
    - <http://www.clker.com/clipart-tonometer-gold-pressure-meter-2.html>