Karlsruhe Institute of Technology

Distributed parametric optimization with the Geneva library

Dr. Rüdiger Berlich
GridKa School 2010

Karlsruhe Institute of Technology (KIT)

Questions

Who in this room knows *exactly*
what
parametric optimization
means ?

Questions

Who in this room knows *exactly*
what
evolutionary and swarm algorithms
do ?

Questions

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

Questions

Who in this room
has been to or used
Geneva
?

Setting the scene

Karlsruhe Institute of Technology

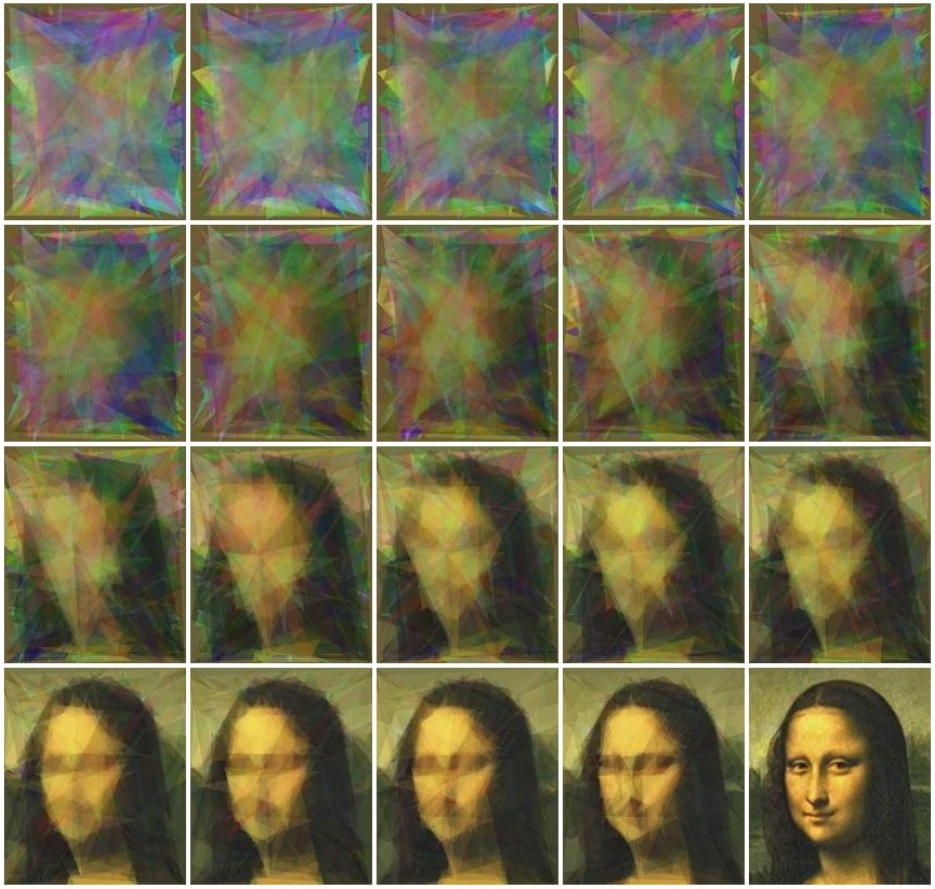
Geneva

(**G**rid-**en**abled **ev**olutionary **a**lgorithms)

- Parallel optimization of problems from scientific and industrial domains
- Covering multi-core machines, clusters, Grids and Clouds
- Implemented in portable C++ (usage of ext. libraries limited to Boost)
- Version 0.82 will be released *today (see http://launchpad.net/geneva)*

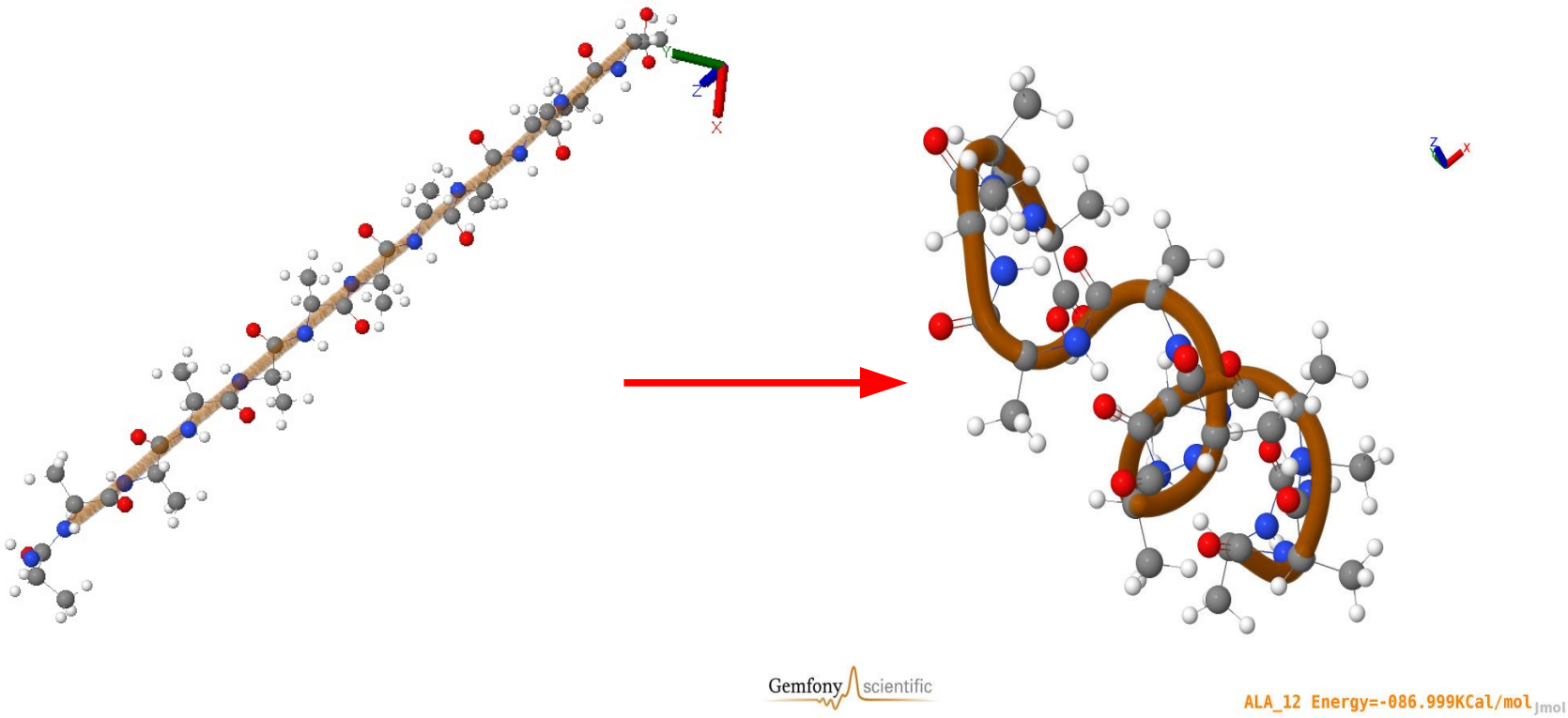
Optimization problems (1)

- **Modelling the Mona Lisa**



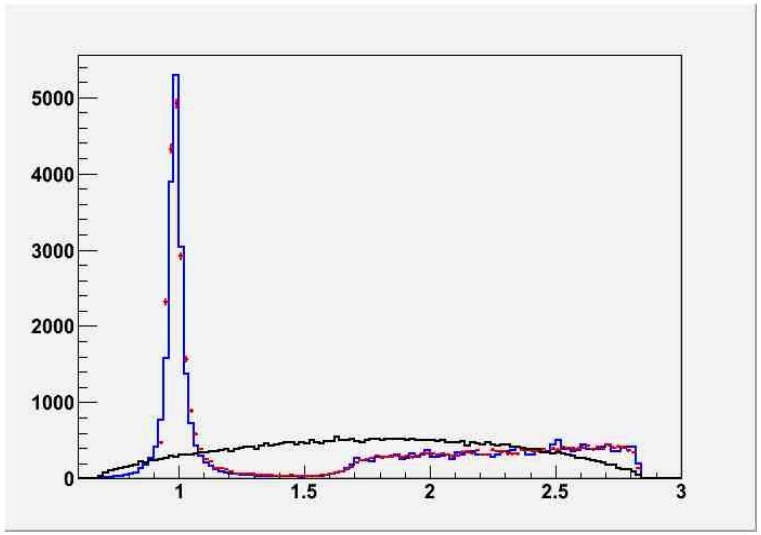
- **Subject of the optimization:**
  - **Alpha-channel, coordinates and colors of 300 triangles**
  - **Means that suitable values for 3000 variables must be found, with no known start-value**
  - **Triangles should be super-imposed in such a way that they resemble the Mona Lisa**

Optimization problems (2)
Protein Folding

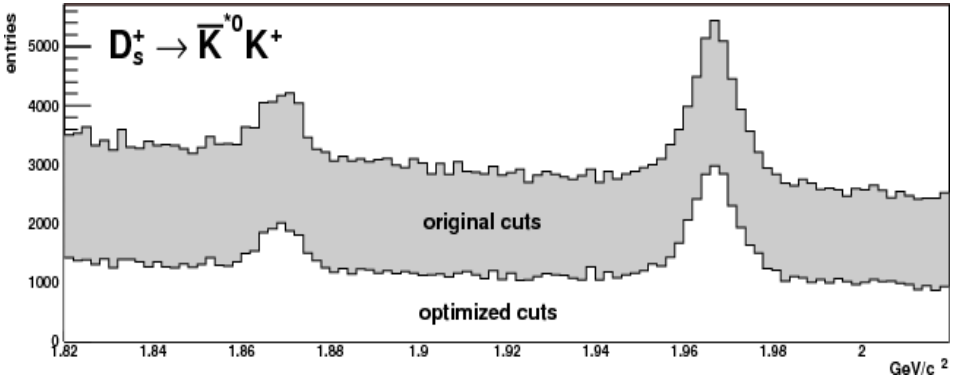


Plots created with the Jmol molecular viewer

Optimization problems (3)
Particle Physics

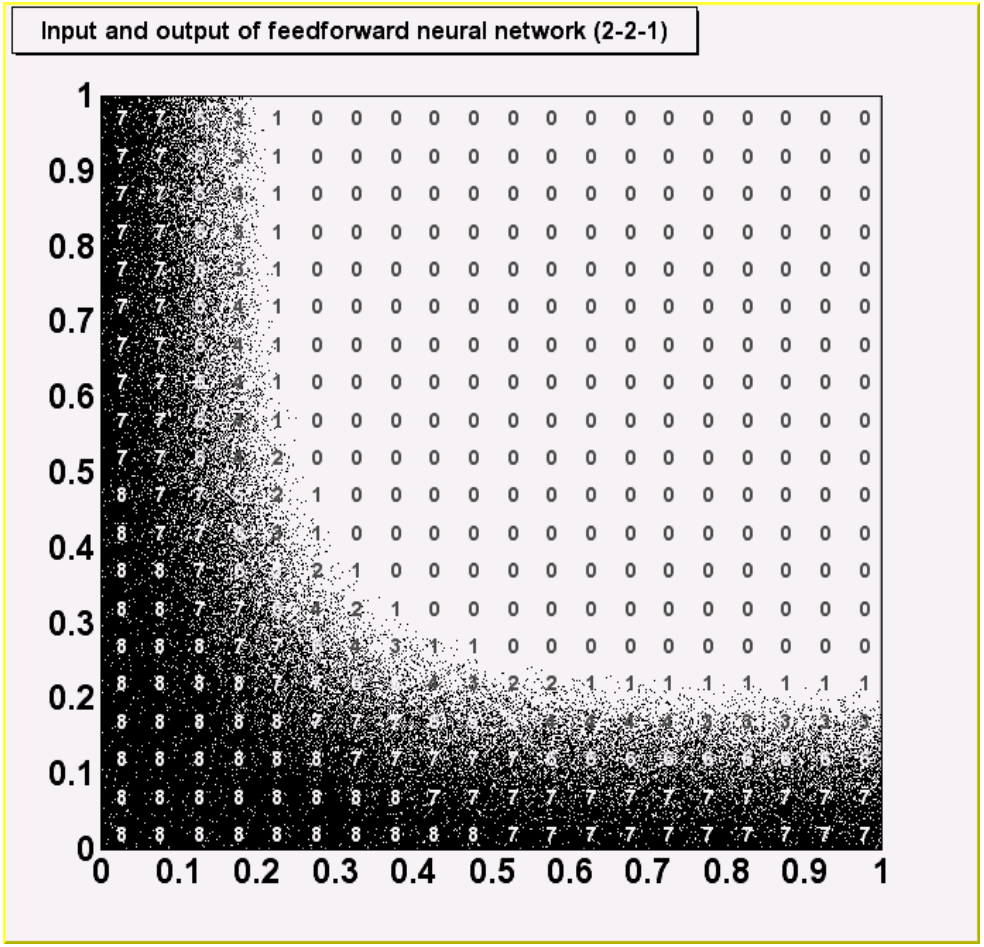


Partial wave analysis



Optimizing an entire particle phsics analysis

Optimization problems (4): Neural Networks



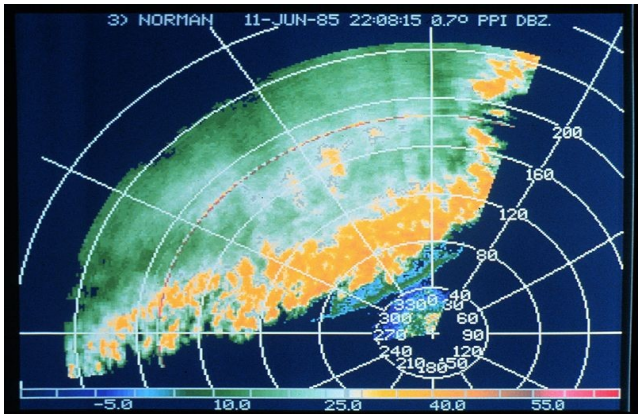
Minimizing the error function of a feed forward neural network is a typical optimization problem.

Shown here:
- Two overlapping data distributions needed to be distinguished
- The output values of the trained network are printed On top of the data distribution
- It is visible that the network achieves an almost optimal separation

Other use cases

- Looking for technical use cases, such as:
  - Optimization of combustion engines
  - Simultaneous calibration of large amounts of parameters
    - Calibration of parameters in the crystal matrix of PP experiments
  - Optimization of „const. parameters“ in simulations (weather, social, …)
  - …

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



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

Our assumption

- **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 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 assumption that performing such computations is not feasible**

What do we want ?

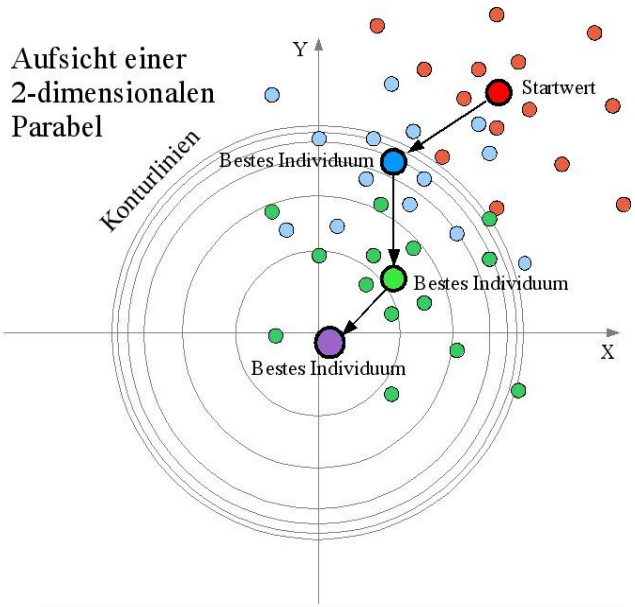
- **Geneva is an Open Source project**
  - **We are building a community**
  - **We are looking for interesting use cases from science and industry**
- **Covered by the Affero GPL v3**
- Code available from http://www.launchpad.net/geneva
- Will be the basis of a spin-off from Karlsruhe Institute of Technology
  - Further information at http://www.gemfony.com

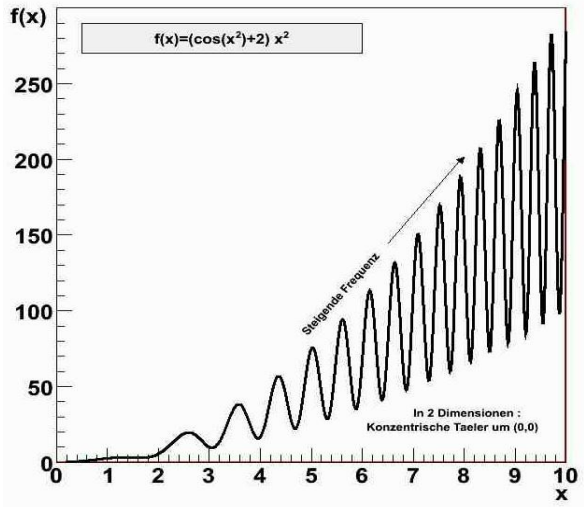
Defining the term „optimization“

- **Realistic approach:**
  - **Optimization refers to the search for the *best achievable result* under a set of constraints**
  - **In comparison: „The ideal“ solution is the *best possible result***
    - ***Usually not practical: Imagine 3000 parameters, test 2 values each. Means computation of 2^{3000} parameter sets***
- **Strategy:**
  - **Identify all relevant parameters, including constraints**
  - **Assign a (computable) evaluation criterion to the parameters**
    - **Encapsulates experts knowledge**
  - **Search for maxima and minima of the criterion using one of many different optimization algorithms**
    - **Generic approach, applicable to many different problem domains**

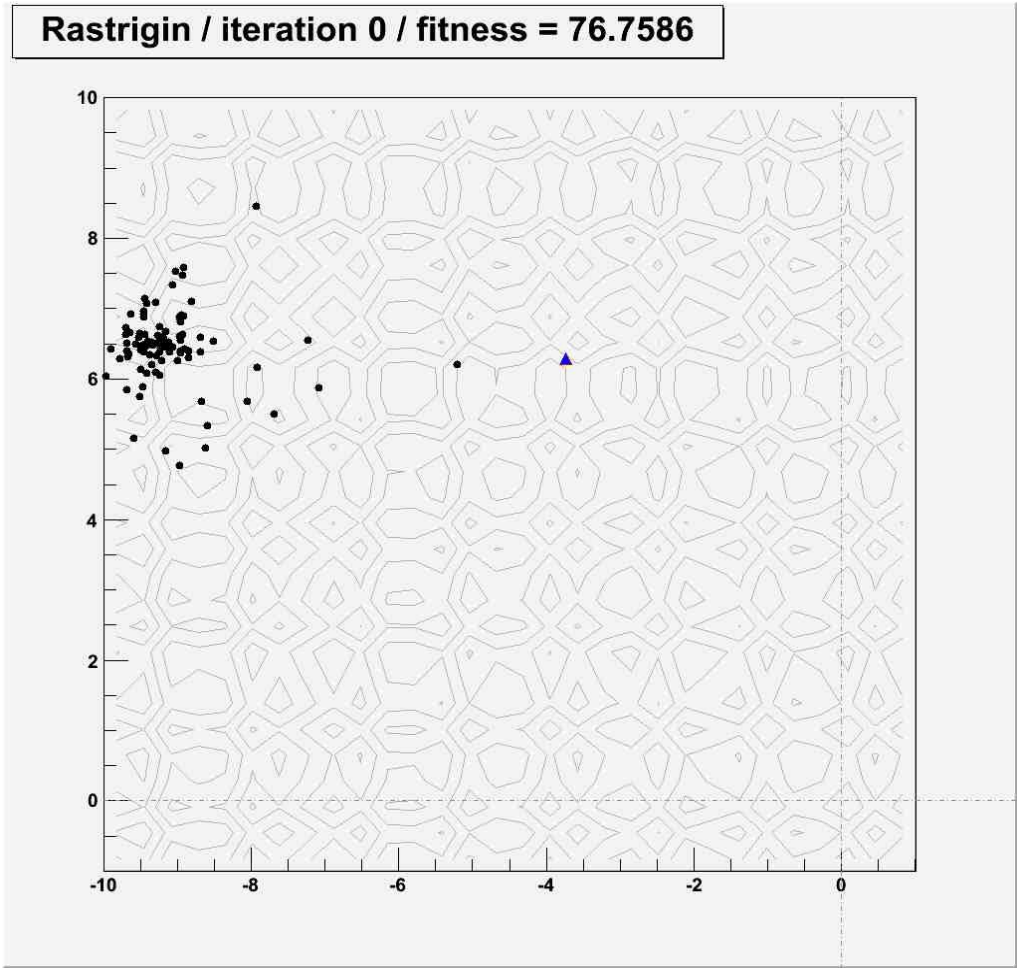
Optimization algorithms: Evolutionary strategies

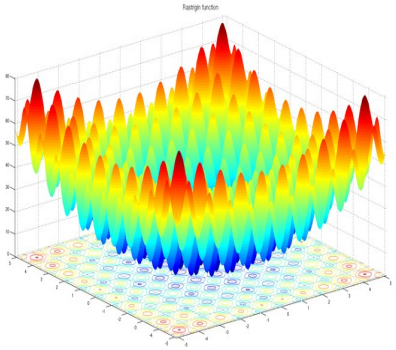
- Algorithm:
  - Population of parents (best known solutions) and children
  - Cycle of duplication, mutation, selection
  - Mutation usually through addition of gaussian-distributed random numbers
- Advantages:
  - Tolerant wrt. local optima
  - Compute time scales with size of the population
  - Easy to parallelise
- Disadvantages
  - Can be slower than gradient descent for smaller problems
  - Many configuration options (e.g. width of gaussian)



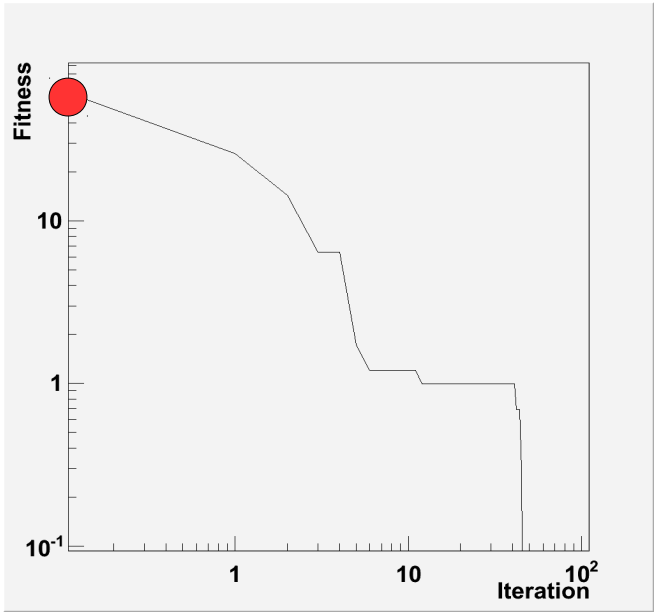


Evolutionary Algorithms: Minimizing the Rastrigin function



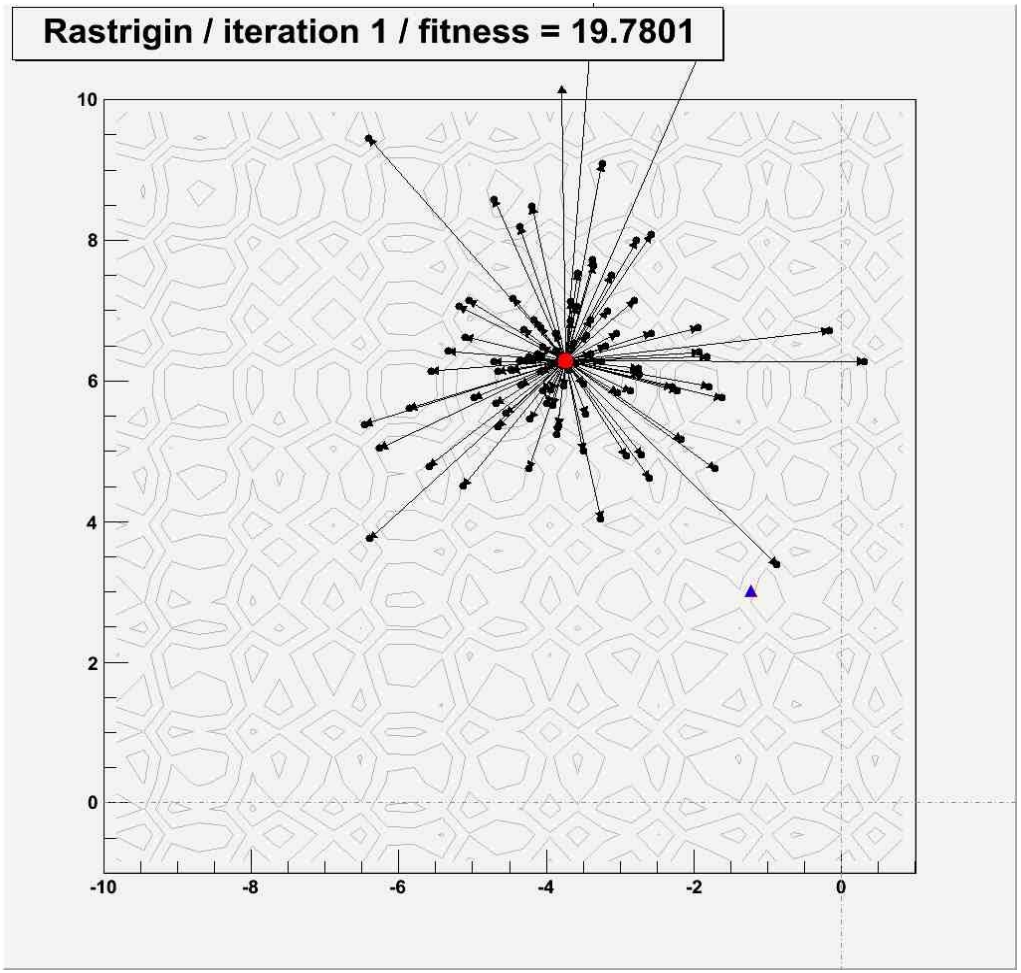


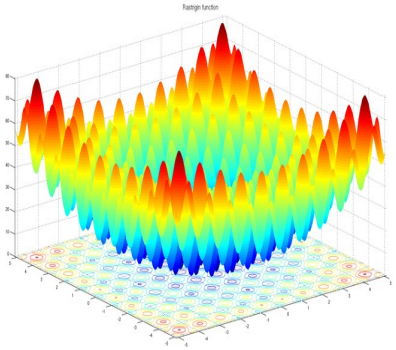
Picture: Wikipedia (public domain)



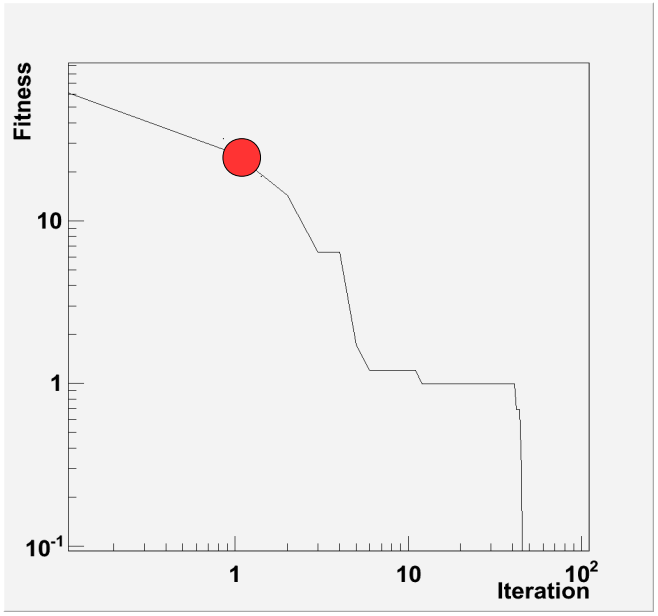
Done with Geneva; Plot created with the ROOT framework

Evolutionary Algorithms: Minimizing the Rastrigin function



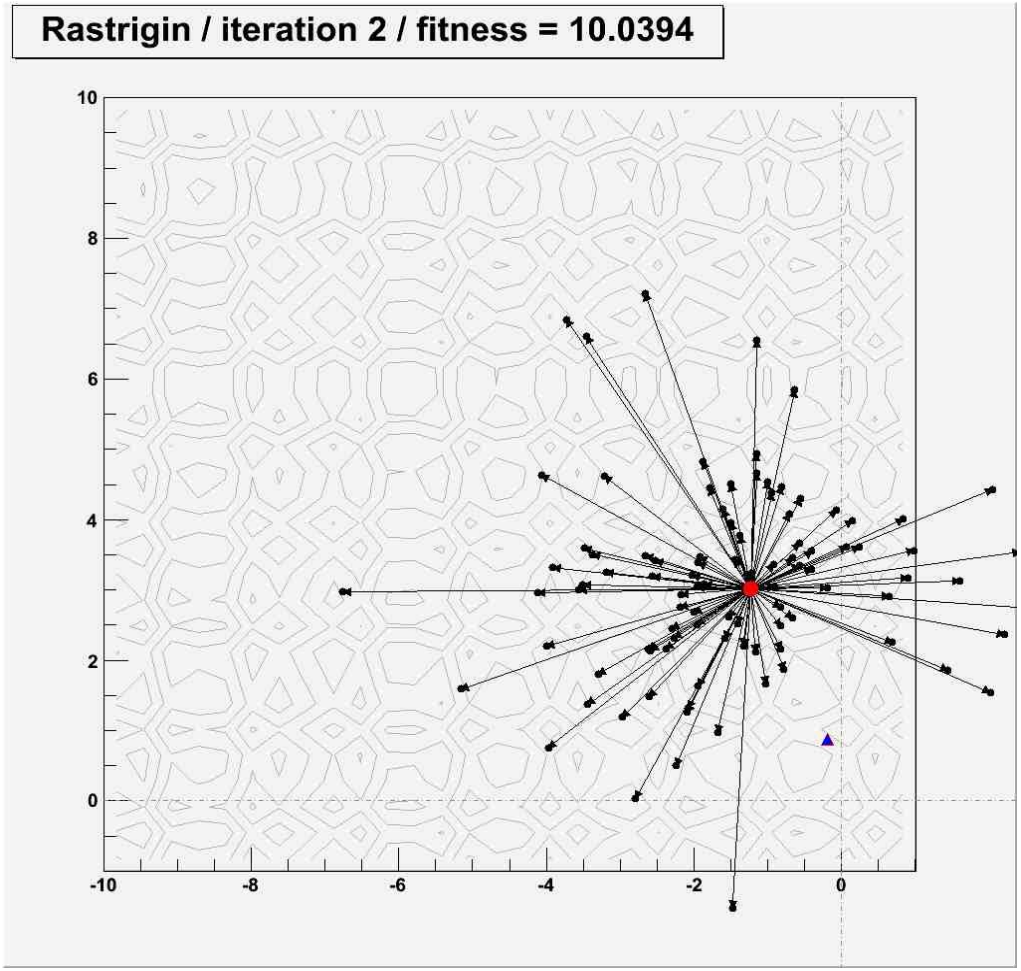


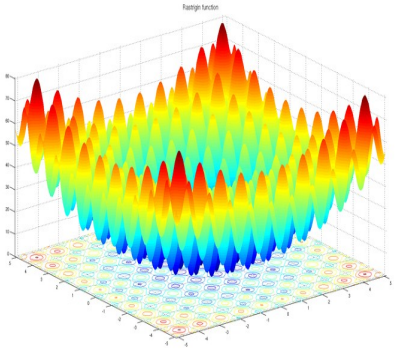
Picture: Wikipedia (public domain)



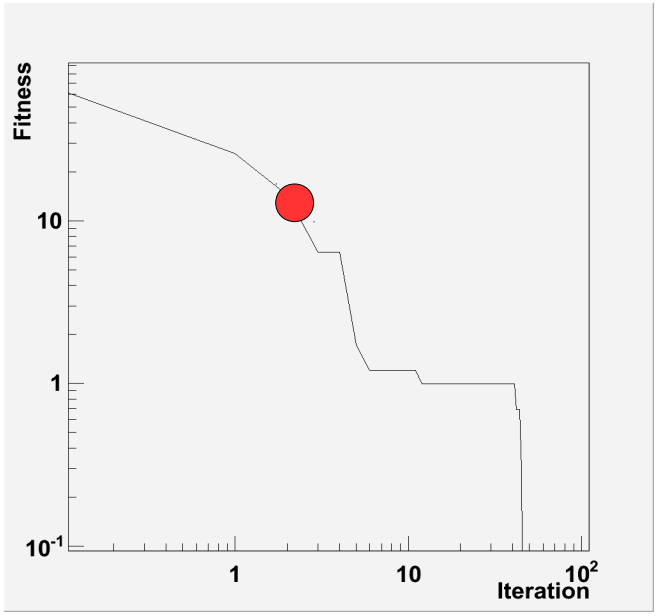
Done with Geneva; Plot created with the ROOT framework

Evolutionary Algorithms: Minimizing the Rastrigin functior



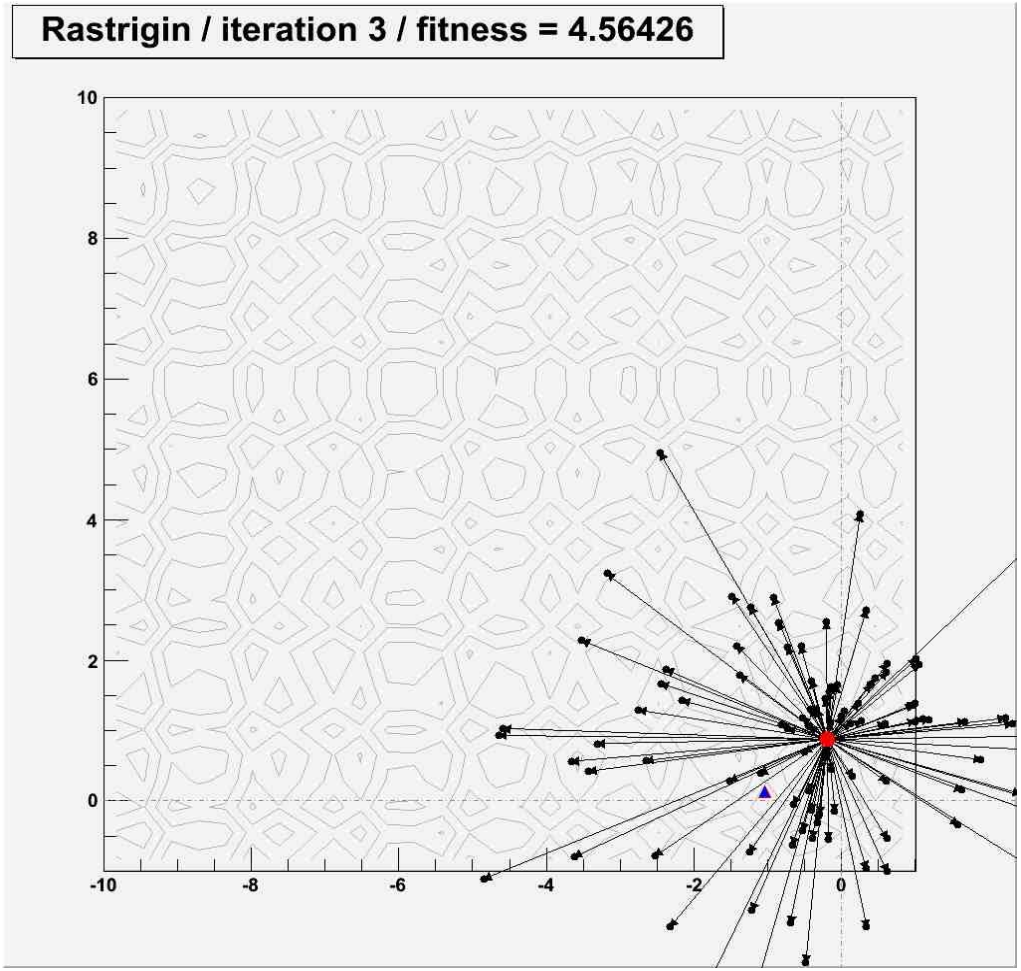


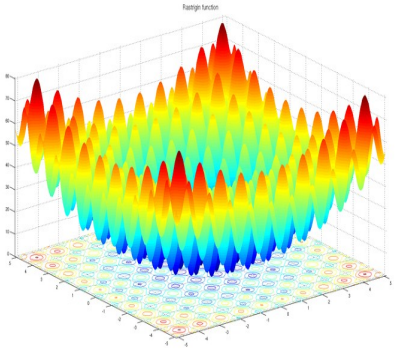
Picture: Wikipedia (public domain)



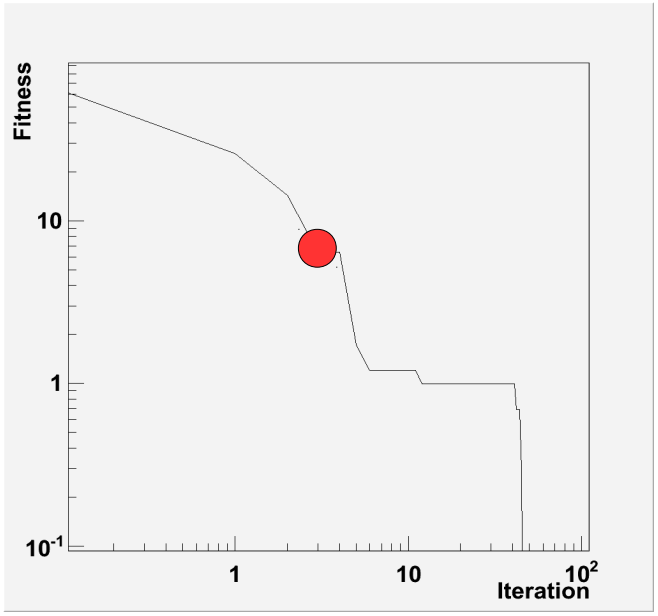
Done with Geneva; Plot created with the ROOT framework

Evolutionary Algorithms: Minimizing the Rastrigin functior





Picture: Wikipedia (public domain)



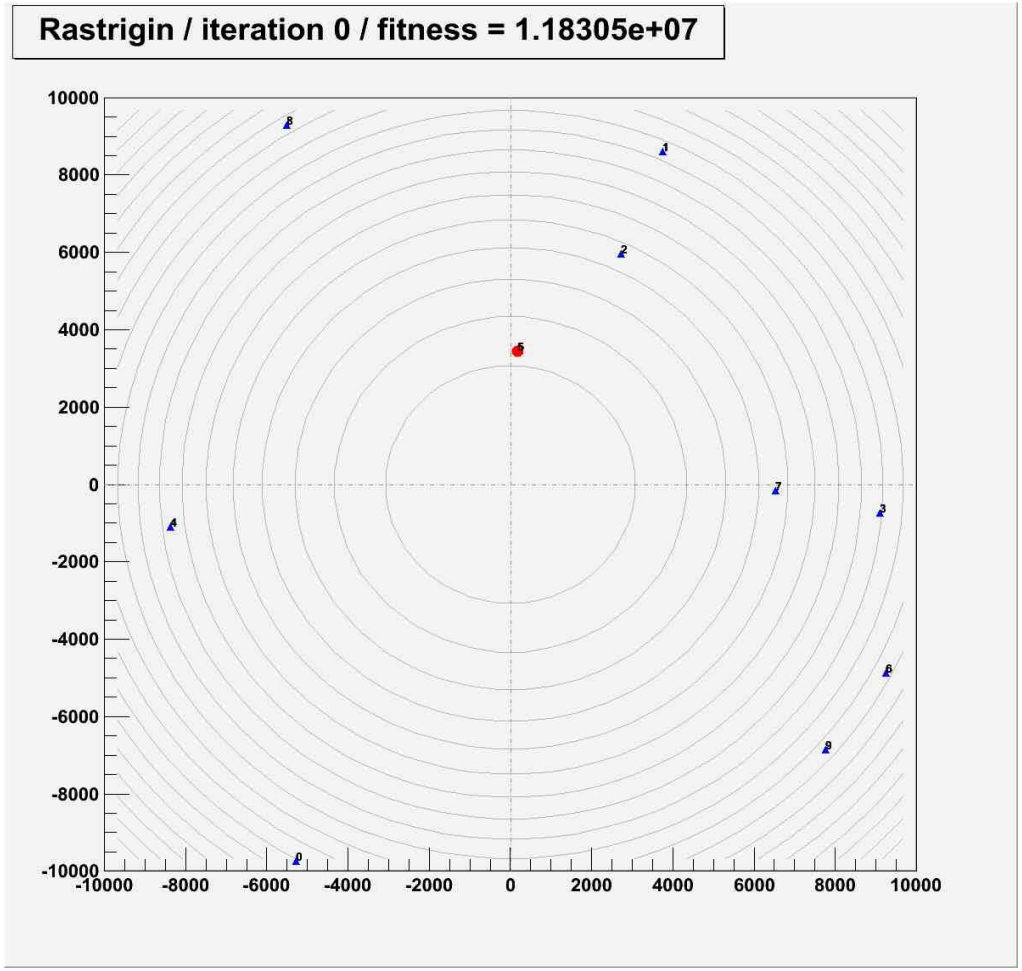
Done with Geneva; Plot created with the ROOT framework

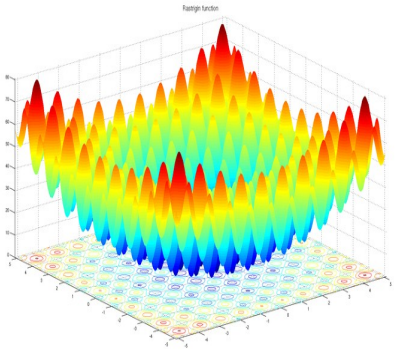
Other optimization algorithms

- 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
  - Swarm algorithms have recently been added to Geneva
- Further interesting algorithms:
  - Gradient descents
  - Deluge algorithms
  - Line search, Simplex, …
- Gradient Descents will soon be implemented in Geneva

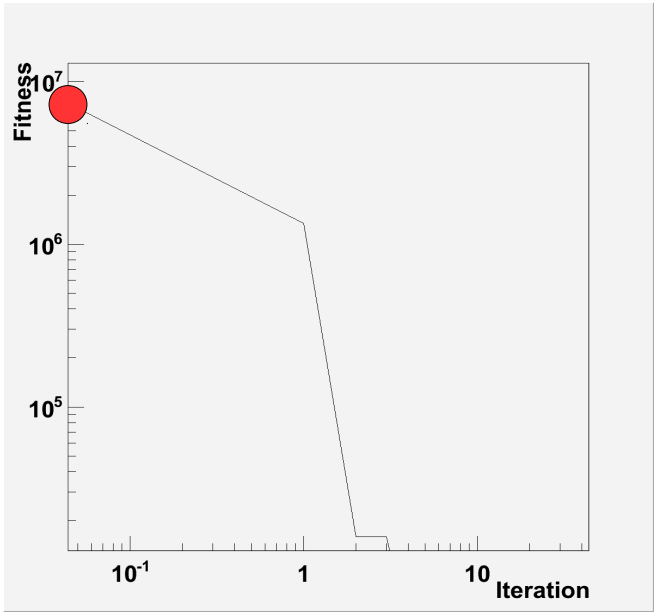
(Source: Wikipedia; Author Mila Zinkova; published under the Creative Commons license „Namensnennung-Weitergabe unter gleichen Bedingungen 3.0 Unported“)

Swarm Algorithms: Minimizing the Rastrigin functior



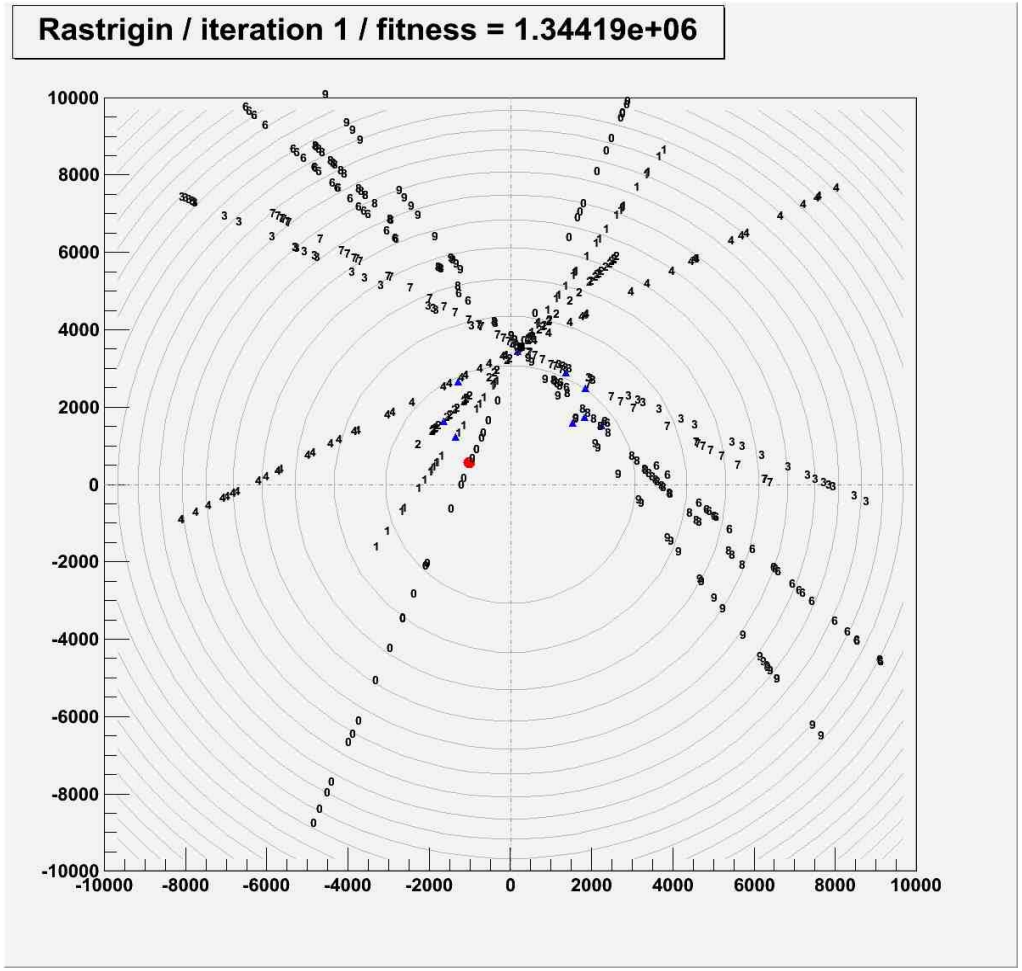


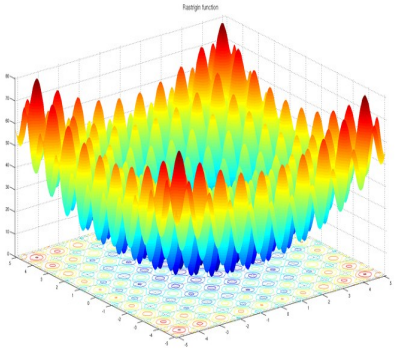
Picture: Wikipedia (public domain)



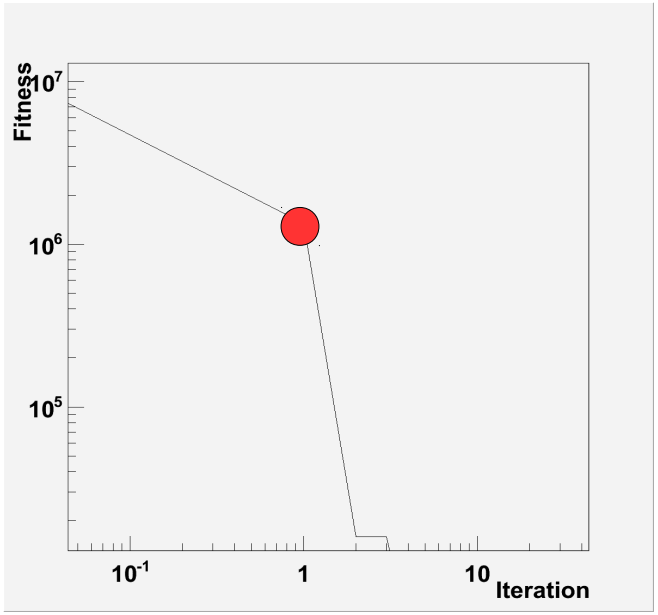
Done with Geneva; Plot created with the ROOT framework

Swarm Algorithms: Minimizing the Rastrigin functior



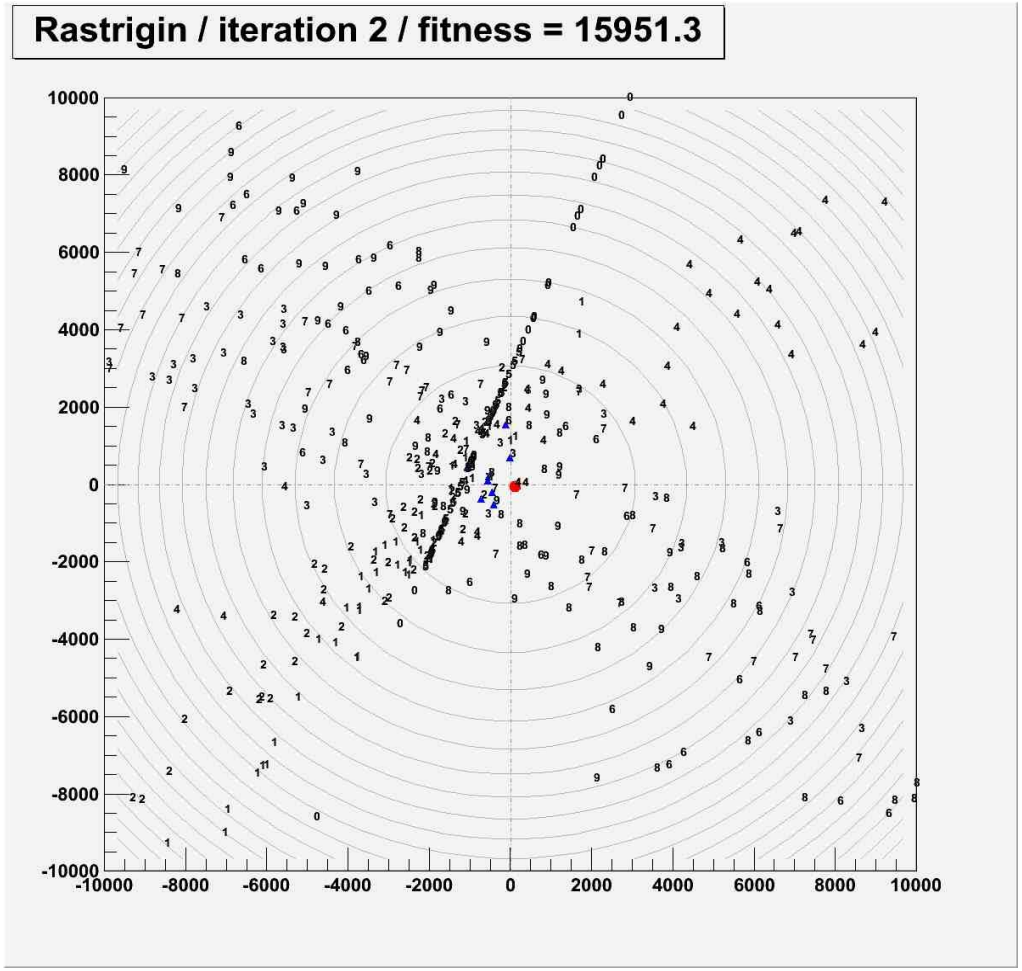


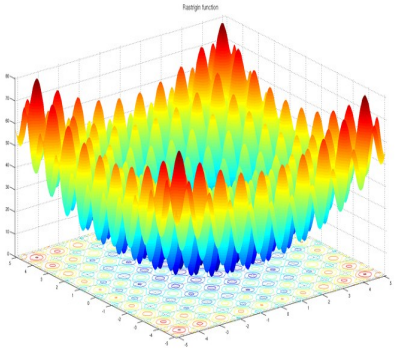
Picture: Wikipedia (public domain)



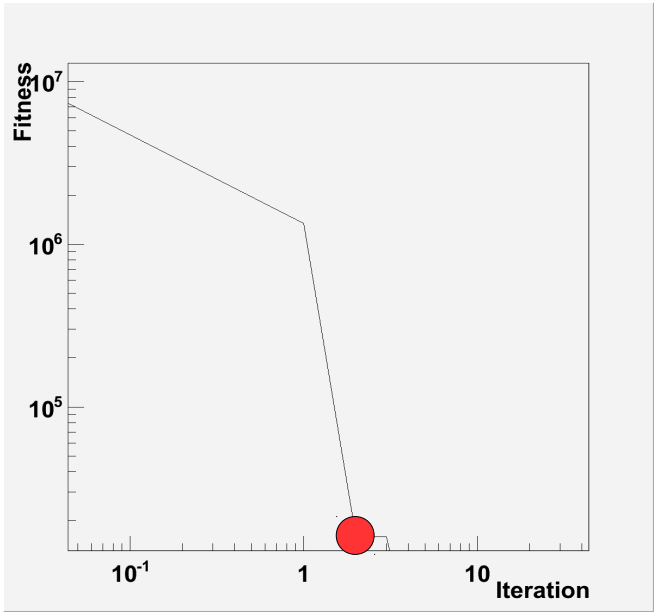
Done with Geneva; Plot created with the ROOT framework

Swarm Algorithms: Minimizing the Rastrigin functior



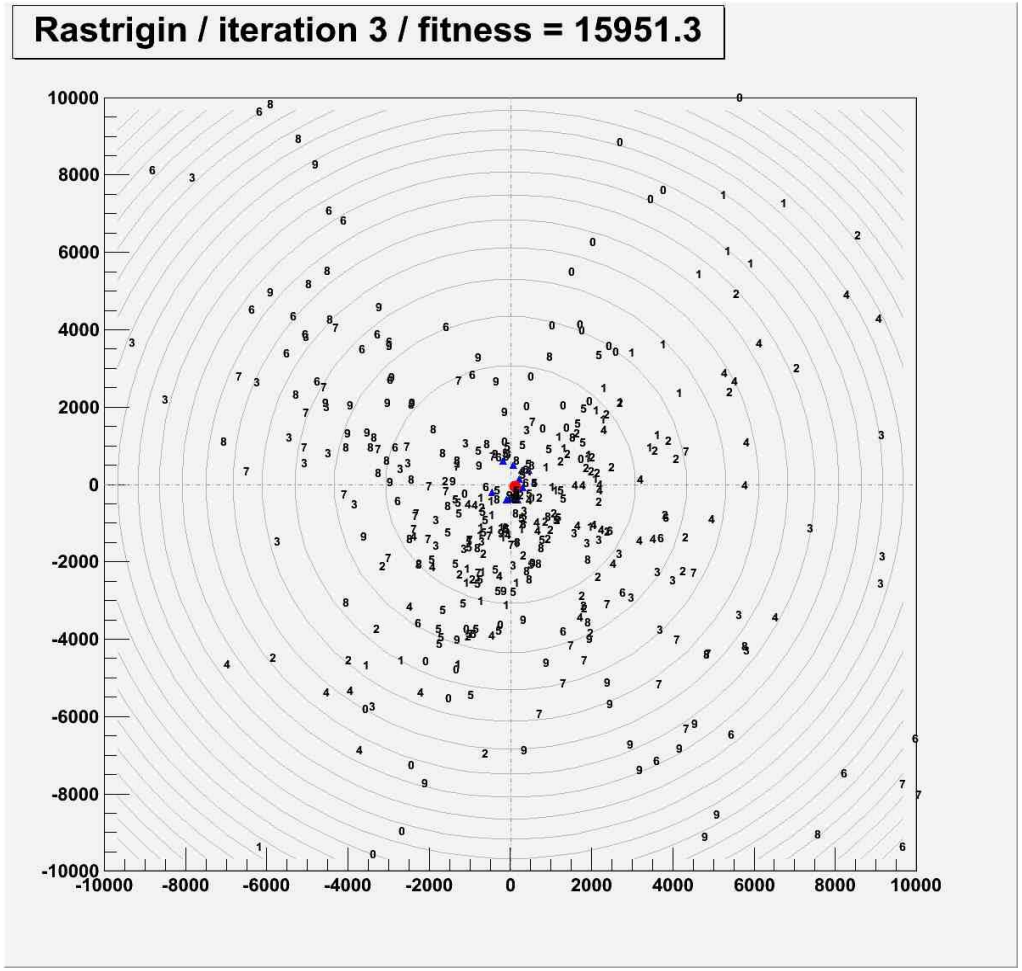


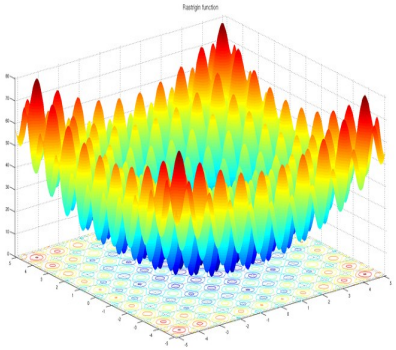
Picture: Wikipedia (public domain)



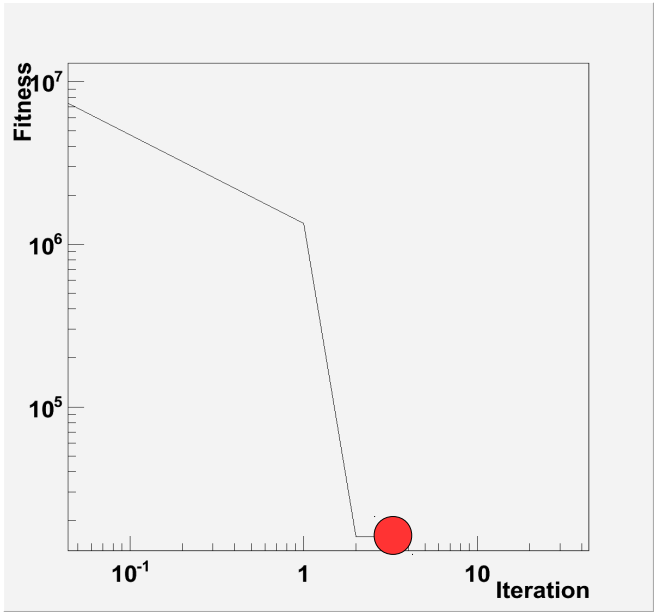
Done with Geneva; Plot created with the ROOT framework

Swarm Algorithms: Minimizing the Rastrigin functior



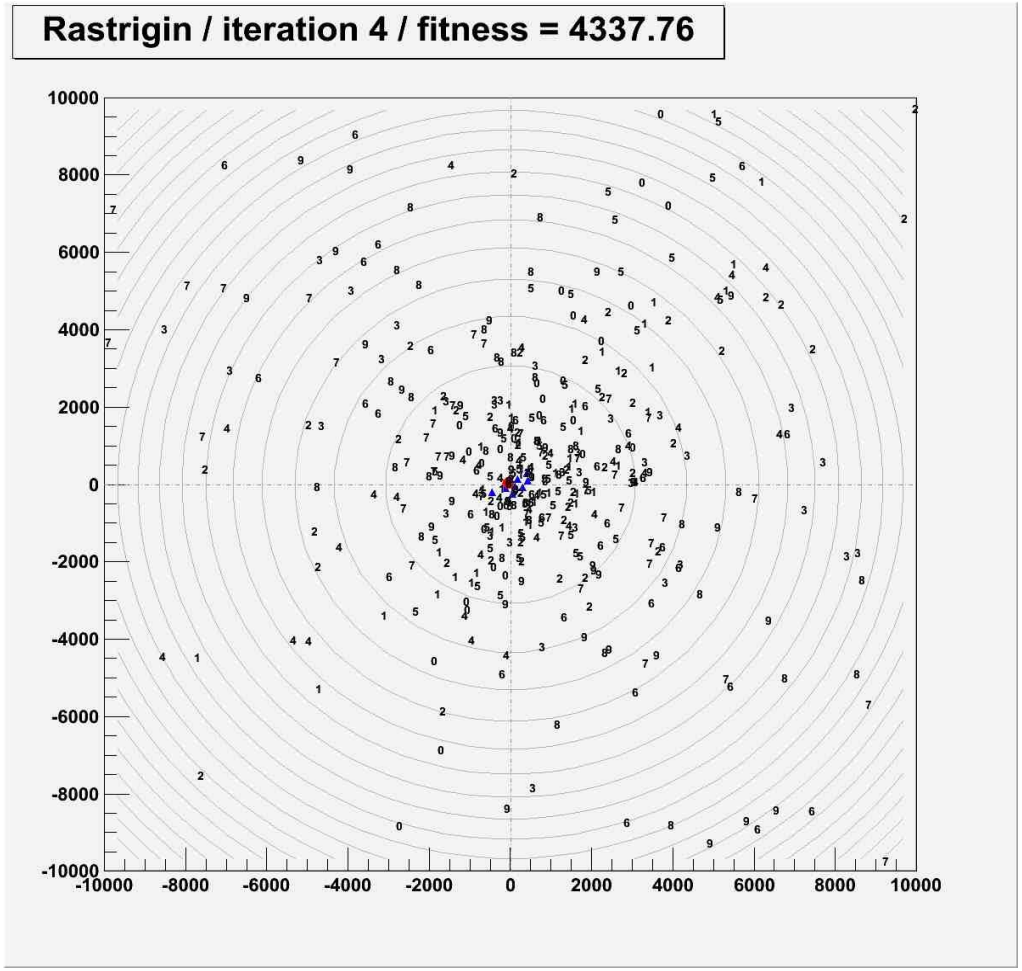


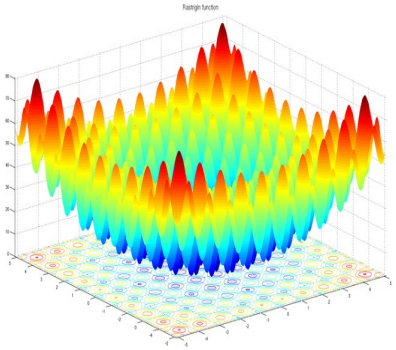
Picture: Wikipedia (public domain)



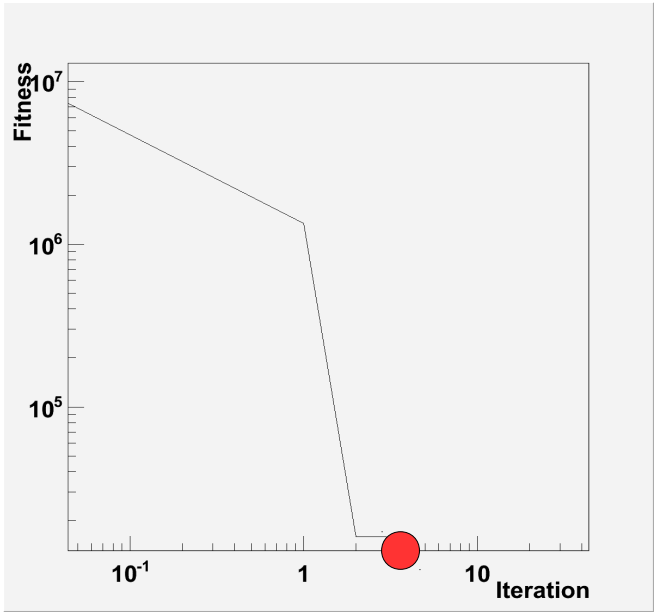
Done with Geneva; Plot created with the ROOT framework

Swarm Algorithms: Minimizing the Rastrigin functior





Picture: Wikipedia (public domain)



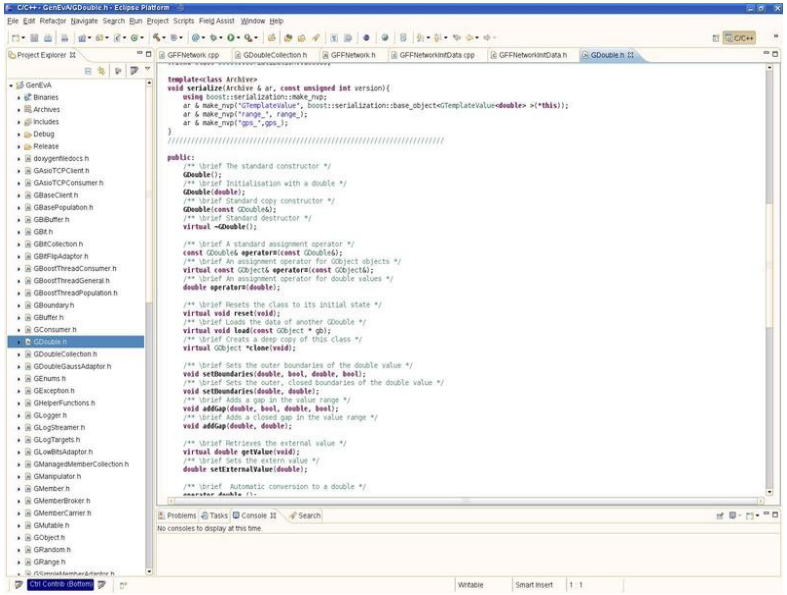
Done with Geneva; Plot created with the ROOT framework

Design criteria

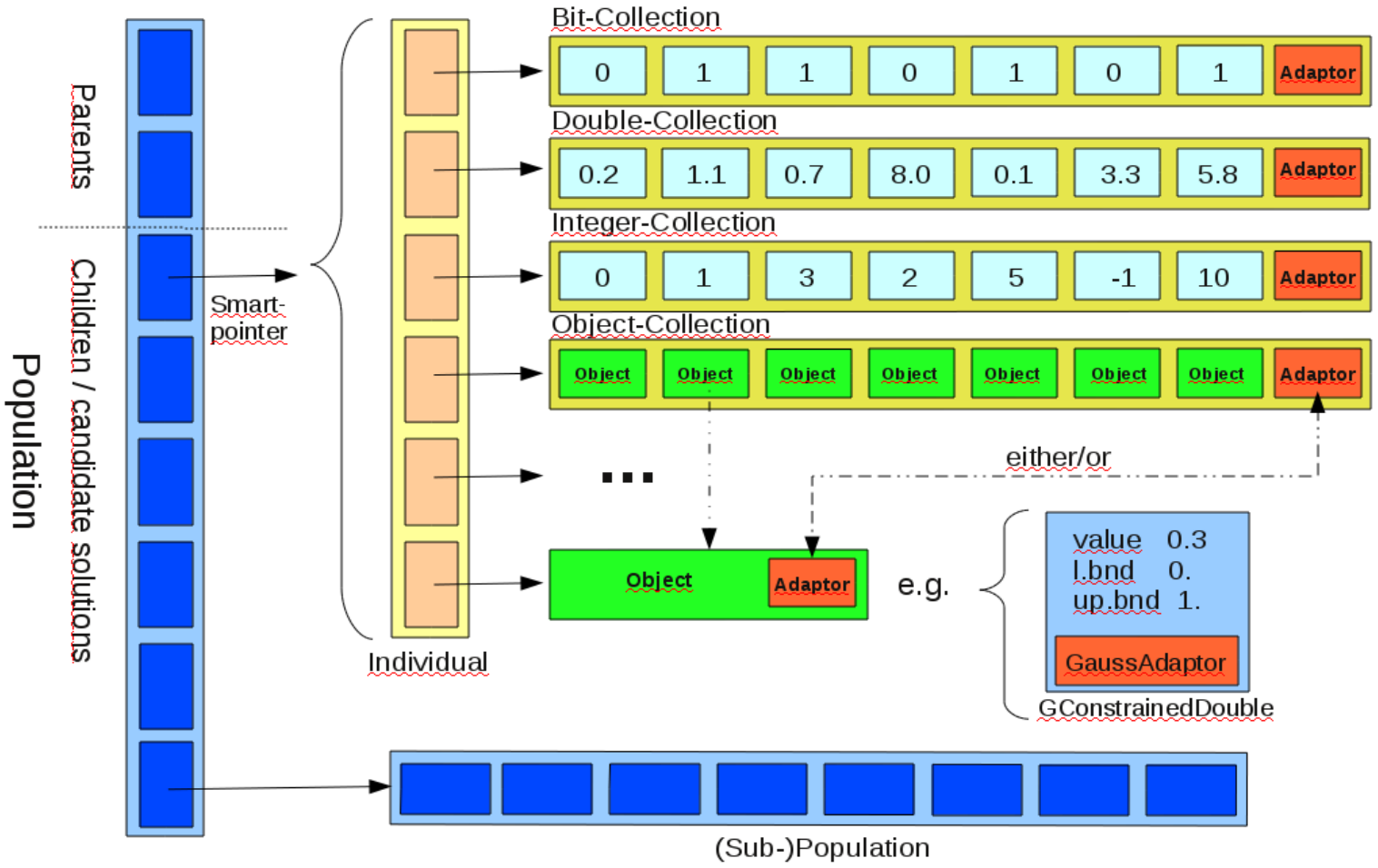
- **Focus on long-lasting, computationally expensive evaluation functions**
  - **Stability *of core library* rated higher than efficiency**
  - **Suitable for distributed environments**
- **Serial, multi-threaded and networked execution, transparent to users**
  - **Implications of networked and multi-threaded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

Implementation

- C++
  - Efficient (cmp. Java)
  - Heavily uses Boost
- So far largely Linux-based
  - But: should be portable
  - Tested with Intel C++, var. g++
- Major components
  - Repres. of parameter sets
  - Optimization framework
  - Parallelization and communication
  - Random number factory



Implementation / Data representation

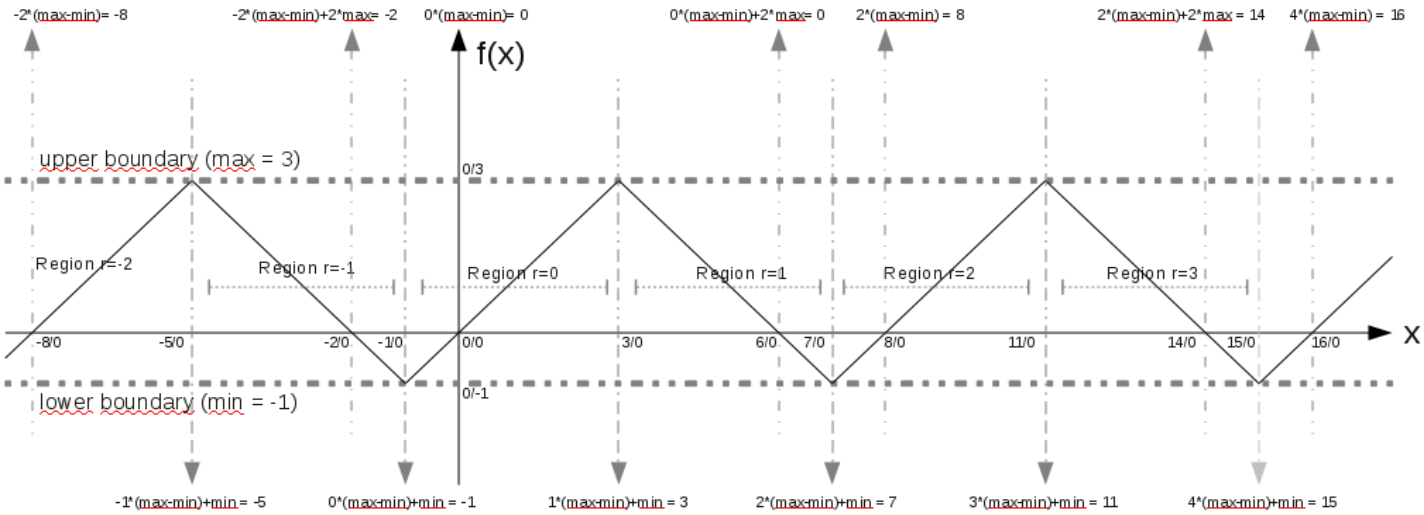


Implementation: Constrained values (e.g. GConstrainedDouble)

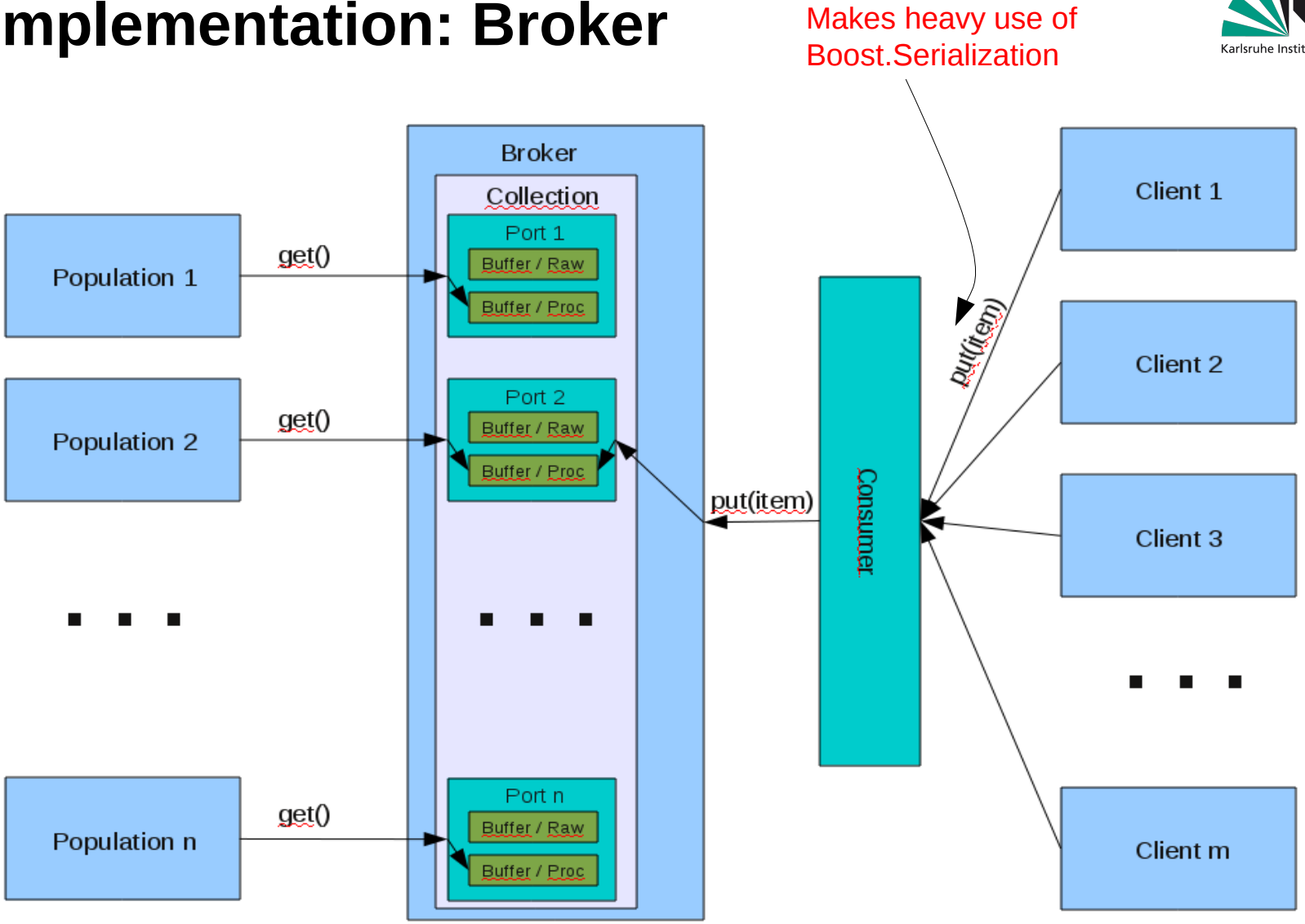
f(x) = x - r*(max-min)

f(x)= -x + ((r-1)*(max-min)+2*max)

with r(x) = floor((x-min)/(max-min))

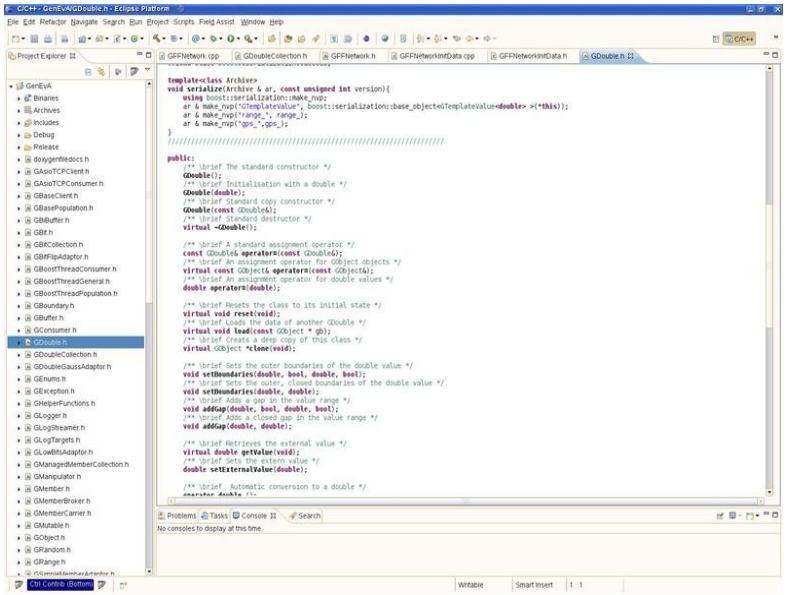


Implementation: Broker

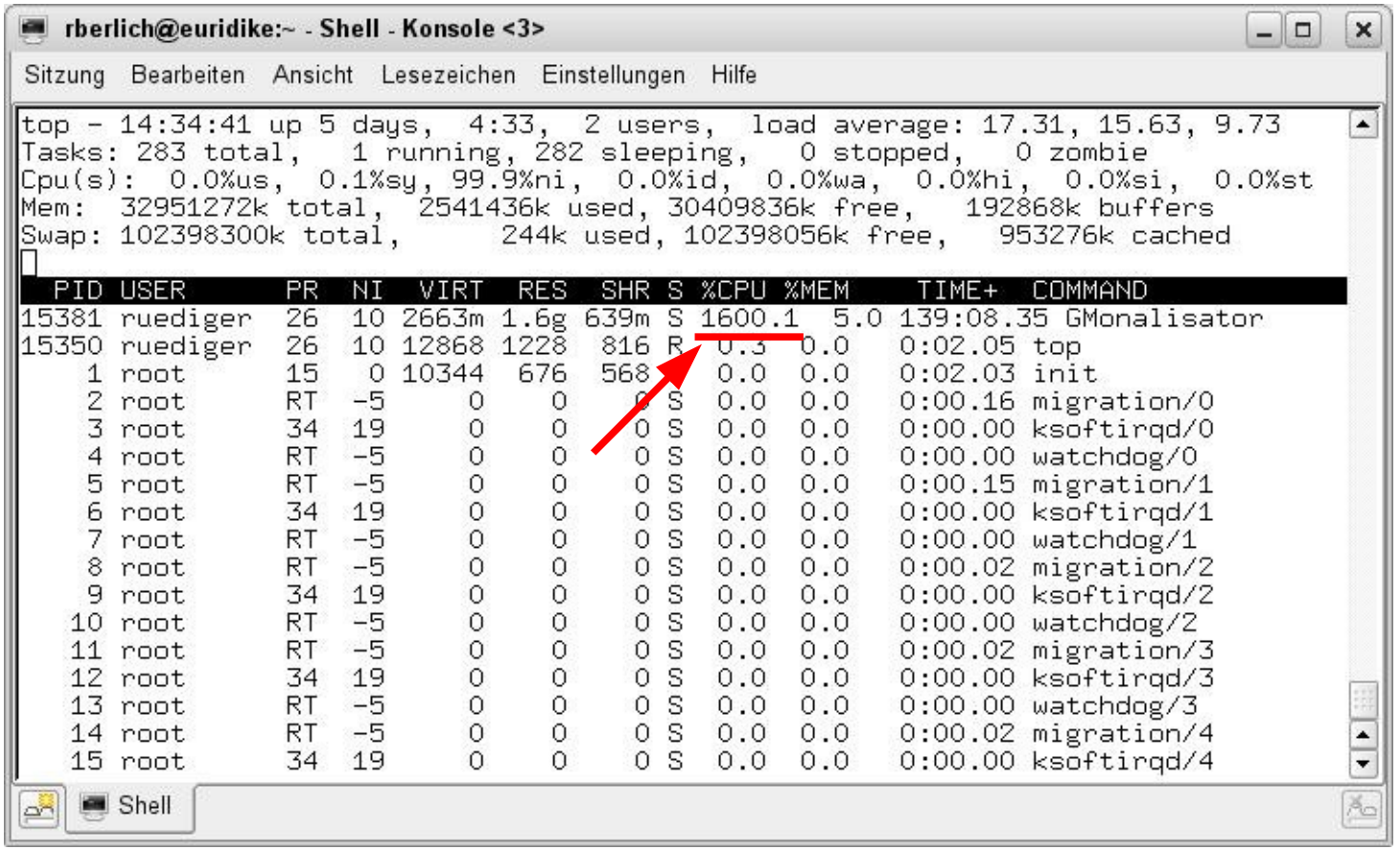


Using the Geneva library

- Code example
  - http://www.launchpad.net/geneva
  - Try: Server and clients on laptop
  - Geneva is a toolkit – need to do some programming to perform optimization
  - Generally: need to specify evaluation function *or* run external evaluation executable
- Running example
  - See examples „GsimpleEA“ and „GSimpleSwarm“, part of the Geneva distribution



Performance

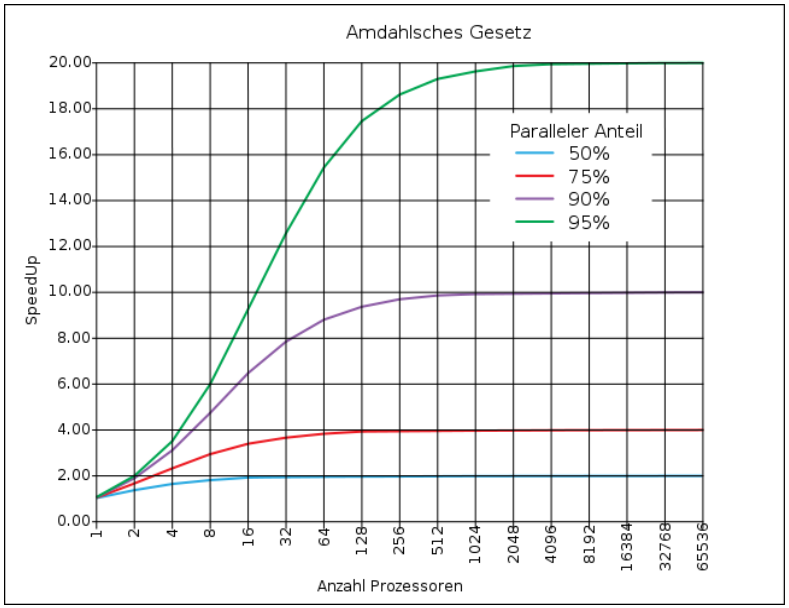


Nehalem system with2 processors / 8 cores / hyperthreading

Performance: Amdahl's Law

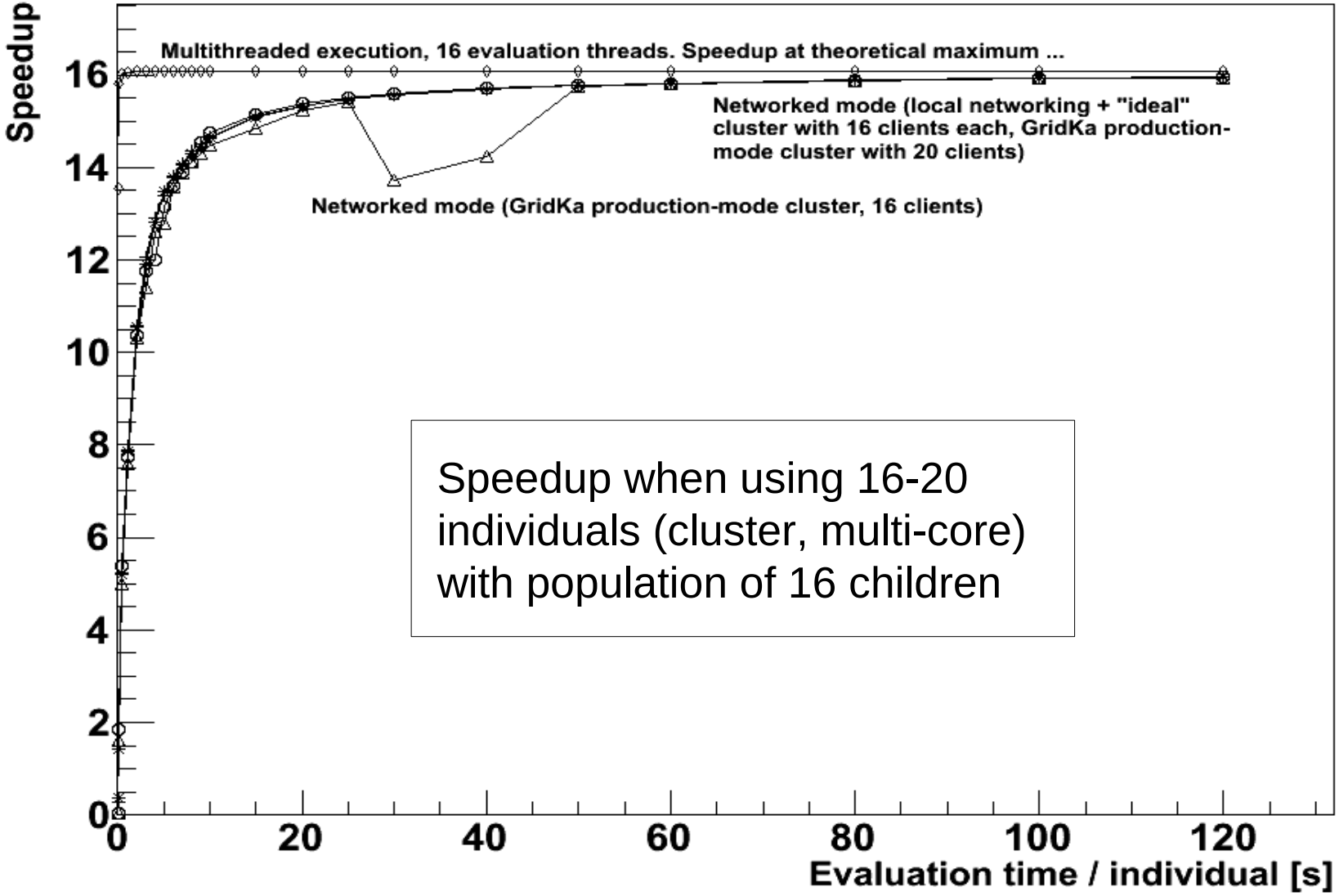
- **Roughly:**
  - **Speedup scales with the percentage of parallel execution time of the overall application runtime**
- **Strong scalability constraints**
  - **Need very high percentage of parallel execution time to achieve significant speedup (as function of the number of parallel processing units)**

Source: http://de.wikipedia.org/wiki/Amdahls_Gesetz
Author of picture: Bob Schwammerl



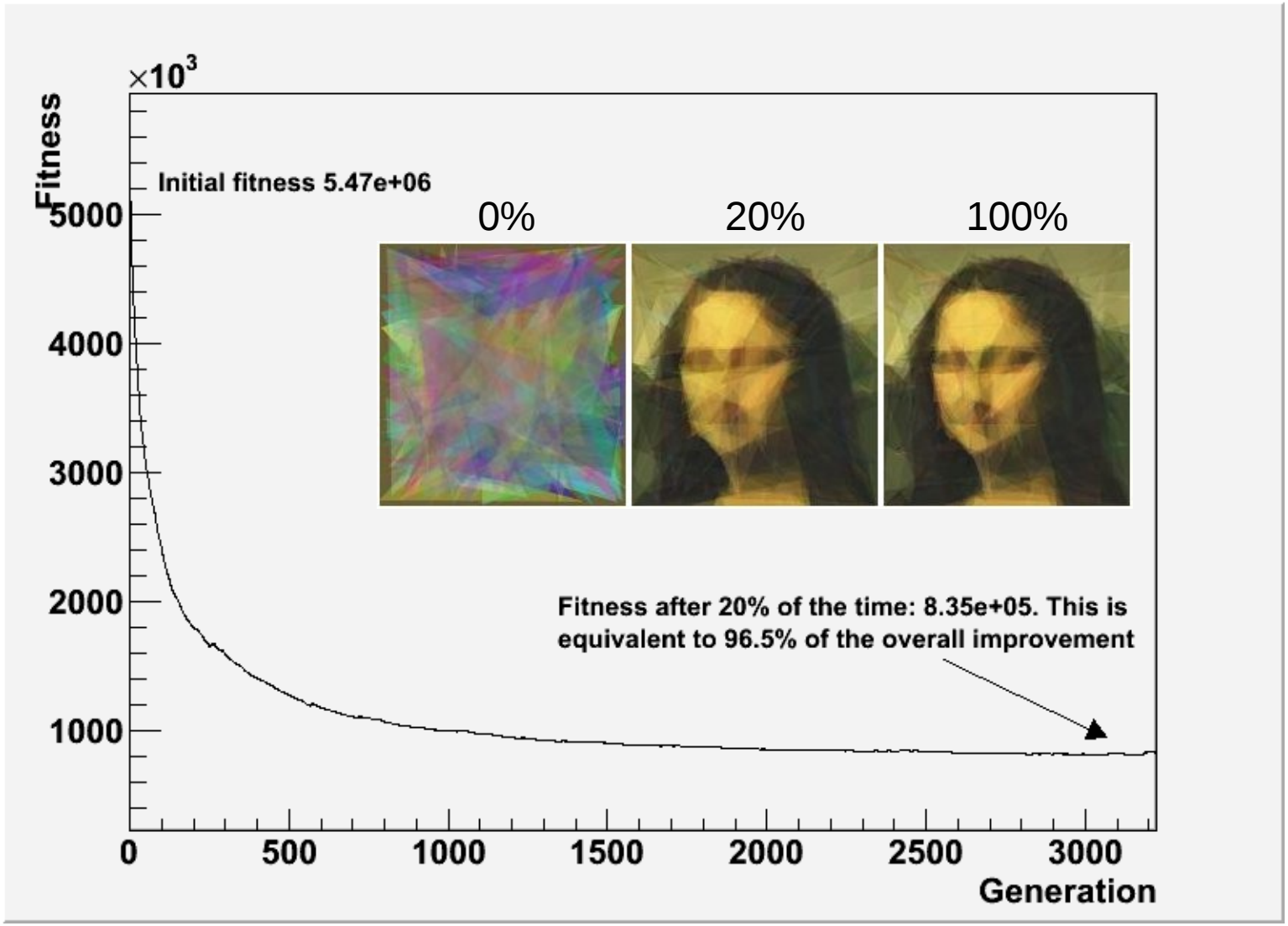
$$S = \frac{1}{(1-P) + o(N) + \frac{P}{N}} \leq \frac{1}{1-P}$$

Performance: Scalability in a network



Speedup when using 16-20 individuals (cluster, multi-core) with population of 16 children

Scalability: Pareto



Moving to a wide-area networking environment (Grid, Cloud)

- Geneva is Client/Server
  - Clients may have a private IP, work in pull mode. Server needs to be reachable, though
  - Server can repair itself in case of a lack of response
  - Late responses will still be considered in later iterations
  - Thus very suitable also for unreliable environments like Clouds
- Must take into account higher latency in WANs
  - Where 15-20 seconds of evaluation time will lead to close-to linear speedup in Cluster, deployment in a cloud environments makes sense for evaluation times beyond approx. 40 seconds (depending on the complexity of individuals – this example: 1000 parameters)
  - We observe „scheduling“ anomalies wrt. network performance similar to http://www.cs.rice.edu/~eugeneng/papers/INFOCOM10-ec2.pdf
- Data management in the cloud can be challenging
- Security is of course better in local clusters
- Otherwise no fundamental difference between cluster deployment and Amazon-style submission of Vms
- (EGEE-style) Grid deployment can be problematic due to very static environment

Summary

- Many low-hanging fruits for distributed optimization both in industry and science
- Deployment in Cluster/Grid/Cloud not only feasible, but highly useful
- Find further information about the Geneva library on http://www.gemfony.com
- Get the software from http://www.launchpad.net/geneva
- We are building a community. Please do contact us with your optimization problems, we are happy to help getting you started with Geneva

Thanks!

- I want to thank the audience and the organizers
- Steinbuch Centre for Computing as well as the department IMA of Karlsruhe Institute of Technology have supported my work – thanks a lot!
- Similarly, I want to thank the Helmholtz Society of German research centres for their kind help
- The Enabling Grids for E-SciencE project has given this work a scientific home for a long time – thanks!!

Question ? Questions!

ruediger.berlich@kit.edu
http://ruediger.berlich.com