Enabling Grids for E-sciencE

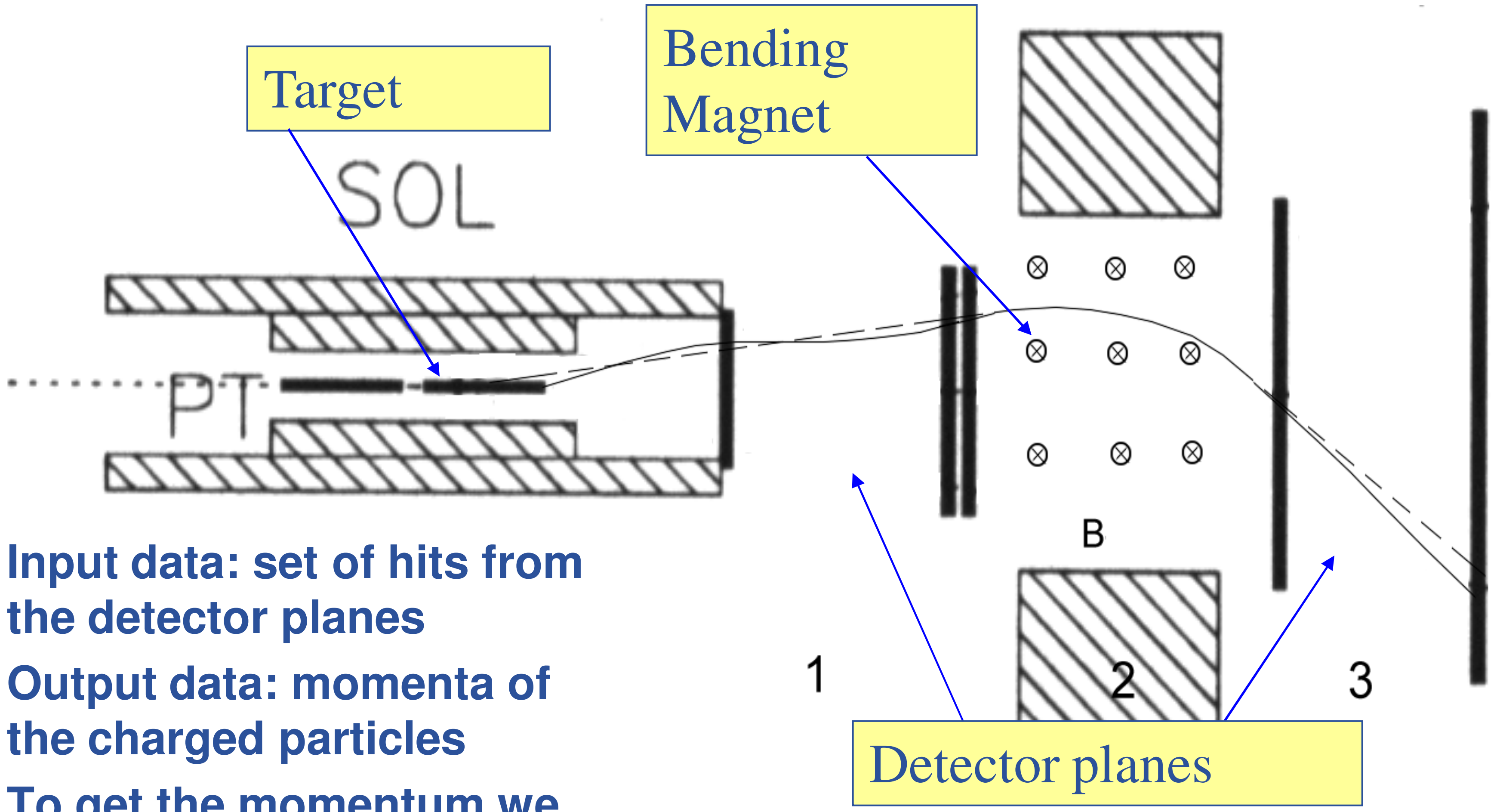
Distributed evolutionary optimization framework on the Grid and HEP use case

Adam Padée, *Krzysztof Zaremba*

University of Warsaw, Warsaw University of Technology

www.eu-egee.org

Example problem – charged particles track reconstruction in COMPASS experiment



- Input data: set of hits from the detector planes
- Output data: momenta of the charged particles
- To get the momentum we need to reconstruct the whole track first.

Problems...

- Particles don’t leave traces in all the detector planes.
- Many hits don’t come from the useful tracks, they are background noise we must filter.
- In one trigger there are tracks from many particles, so we must correctly assign hits to tracks.
- Mathematical models used in reconstruction are simplified and detectors have finite spatial precision.
- To overcome that problems we need to set some arbitrary parameters (geometrical tolerances etc.) that will control the reconstruction process

Optimization – why EA?

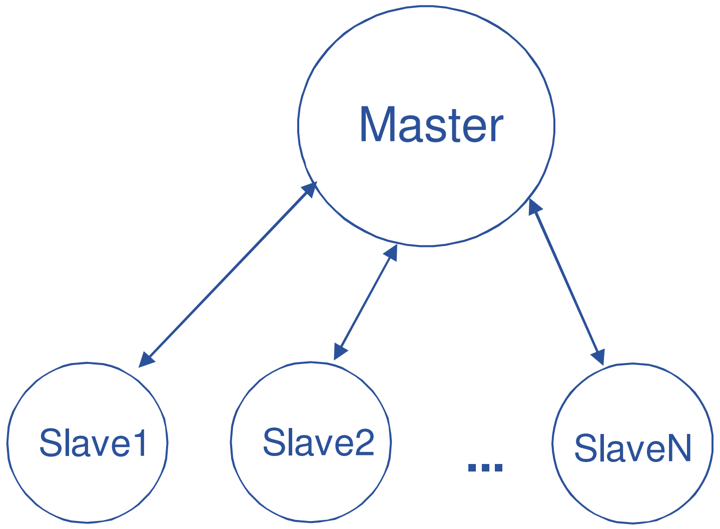
- Tuning of the reconstruction parameters can significantly improve efficiency
- Using Monte-Carlo simulation and measured magnetic filed maps we can check how good is the set we currently have
- There are couple of tens parameters to change
- Evaluation of each parameter set takes at least couple of minutes (on an x86 core running at ~ 2-3GHz)
- Optimization has to be repeated on each detector setup
- Efficiency is expressed in the number of properly reconstructed tracks, so we have many plateaus and local optima on objective function landscape
- We need a method that will not rely directly on gradient information and can be used on distributed resources

How we can use Grid?

- Huge resources
- Difficult communication between processes, making it hard to use the resources in a coordinated way
- Different processing speeds of the Worker Nodes
- Failures
- We must have an algorithm capable of dealing with that

Architectures of the parallel EAs

- **Master-slave**
  - There is one population, and it is stored on a server (master node). Calculation of the fitness function values is distributed among the worker nodes (slaves). We have two general types of this EA:
    - Synchronous
    - Asynchronous (split population or SSEA – Steady State Evolutionary Algorithm)

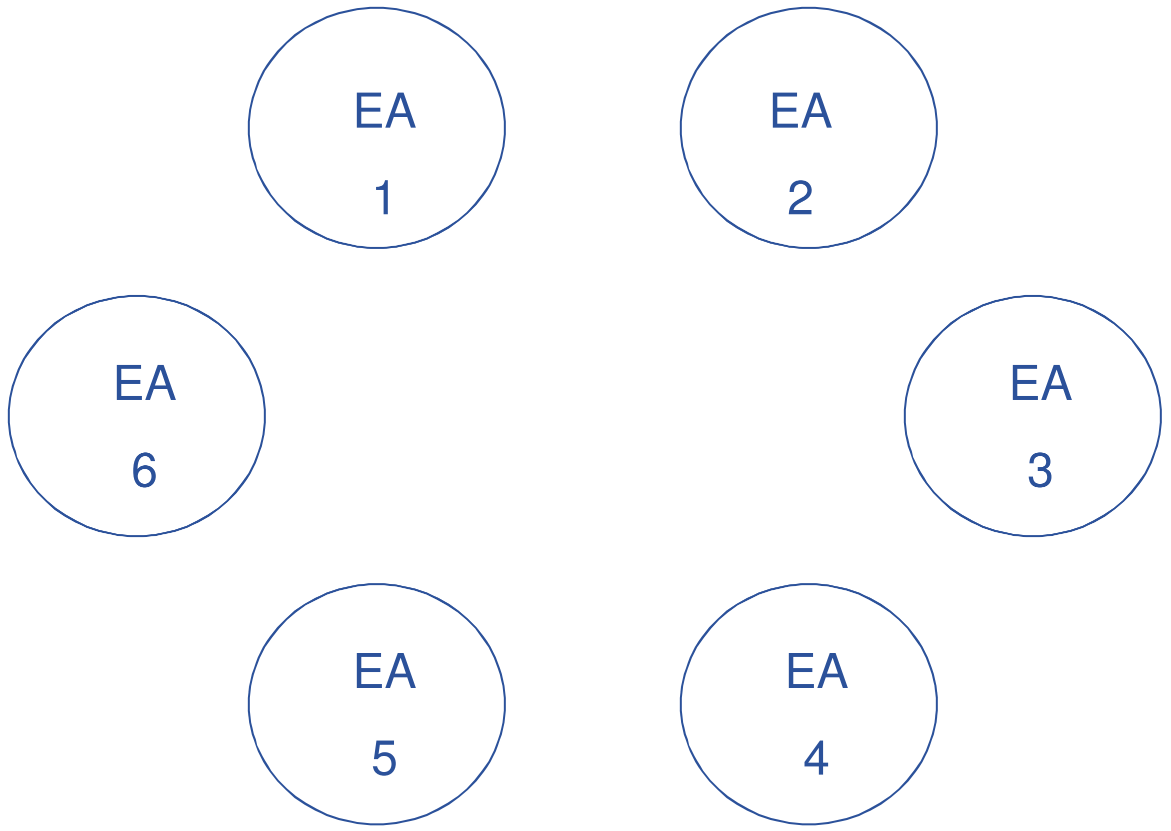


Architectures of the parallel EAs

- **Multiple population algorithms (also called coarse-grained)**
  - They consist of multiple independent populations, called demes, exchanging only selected individuals. Frequency of the exchanges, migration channels and the operators applied to the individuals depend on the model, e.g.:
    - Fully connected topology (suitable especially for parallel supercomputers)
    - Island Model (arbitrary topology, also simple migrations but less frequent)
    - Advanced migration strategies (pollen transmission, social ...)

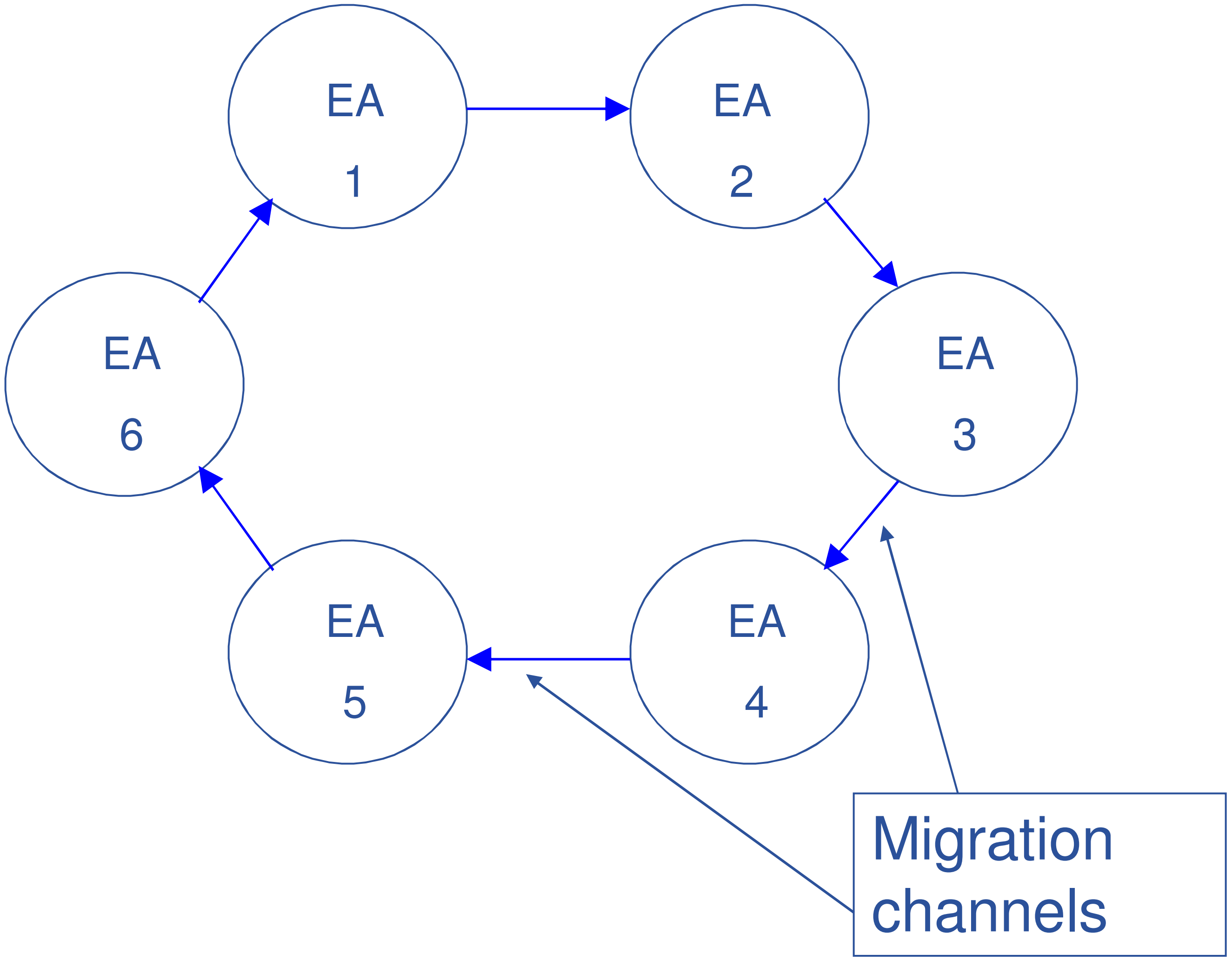
Island Model EA: basic concept

Growth phase: population on each of the islands is being developed independently



Island Model EA: basic concept

Migration phase: Each of the populations selects one or more individuals (usually the best ones) and sends them to the neighboring island, where immigrants are inserted to the local population



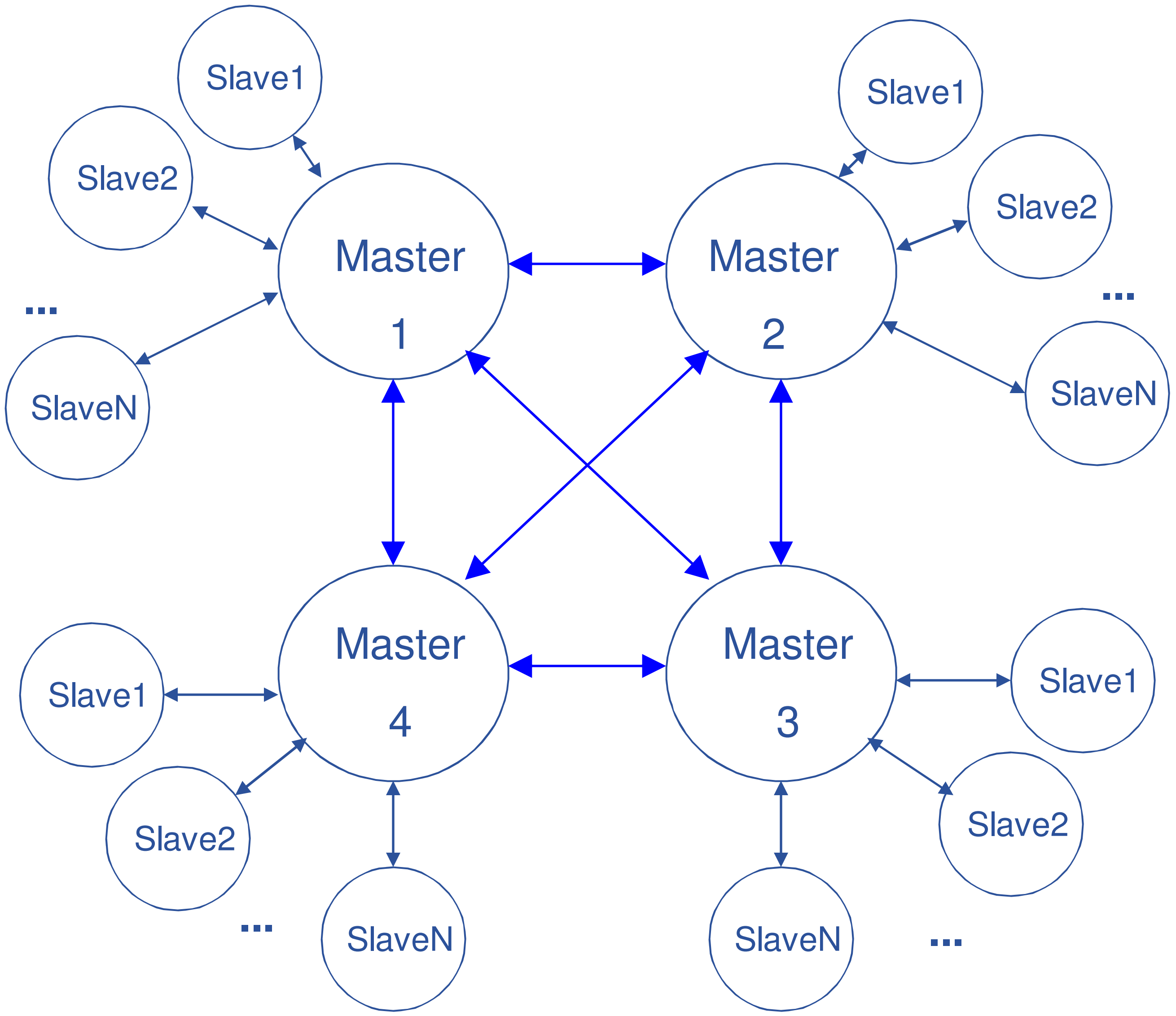
Architectures of the parallel EAs

- **Cellular (also called fine-grained)**
  - One population divided spatially among neighboring processors. Each of them can process one or more individuals. Selection and crossing-over takes place only among neighbors. Most popular implementations:
    - Hardware (dedicated integrated circuits)
    - Software: usually on SIMD processors, although there are also very efficient implementations on ccNUMA architecture (Cache Coherent Non-Uniform Memory Access)

Architectures of the parallel EAs

- **Hybrid**
  - Coarse-grained algorithms consisting of multiple cellular or master-slave algorithms. This is the most advanced, and also most flexible architecture.

Example of hybrid EA



Which is the best on the Grid?

- **Master-slave could be easily implemented submitting one job for each individual, but ...**
  - Submission overhead can reach order of minutes for a single job
  - WMS + L&B is not prepared for a massive submission of short jobs (frequent failures, disturbance for other users)
  - Complicated task monitoring and error analysis
- **Island model EA**
  - One deme per every CPU requires high migration rates (bandwidth problems)
  - Flat model of communication hard to implement across sites.
    - Grid-wide MPI not available in big production grids like EGEE
    - Introduction of dedicated service is laborious and not flexible
    - Possible exchange of information via files on SEs – easy, but very slow and inefficient solution
- **Cellular**
  - Not suitable because resources are not uniform.

Which is the best on the Grid?

- **Hybrid EA**
  - One island can be formed on each cluster.
    - Communication within a cluster is fast, so we can use master-slave algorithm here
  - Relatively big population size at each island allows much lower migration rates than in flat island model
    - Migrants can be exchanged also via files on the SEs
  - In fact it is the best architecture for the Grid, because multiple experiments have shown that the best results can be obtained if communication topology is adapted to the physical structure of the resources

Hybrid algorithm on the Grid

- **Problems ...**
  - To ensure consistent convergence speed, the size of the population at each island should reflect the available number of CPUs at the site.
    - According to (Branke, Kamper, 2004) this shouldn't be a big problem, especially when we set higher migration rates at smaller islands
  - Due to inevitable differences in clusters' processing speeds, all migrations should be done asynchronously
    - Technical difficulty, could be overcome by splitting the migration into two phases: export when an island publishes an individual and import when it downloads individual(s) from other islands

The algorithm

- **Lower level (cluster): Master-slave asynchronous SSEA**
  - Master constantly submits new individuals to the slaves and immediately inserts newly calculated ones to the population (there are no generation changes)
  - There are two variants of internal communication, selectable at runtime by appropriate option
    - MPI (asynchronous messages are used) - RECOMMENDED
    - GFAL – individuals are saved on local SE (NOT recommended)
- **Higher level (Grid): Multi-deme EA with asynchronous migrations**
  - Migrants are recorded on a SE as files – each deme under it's own filename demeXXXX.bin
  - Whenever an island wants to import an individual, it chooses random file belonging to other deme.

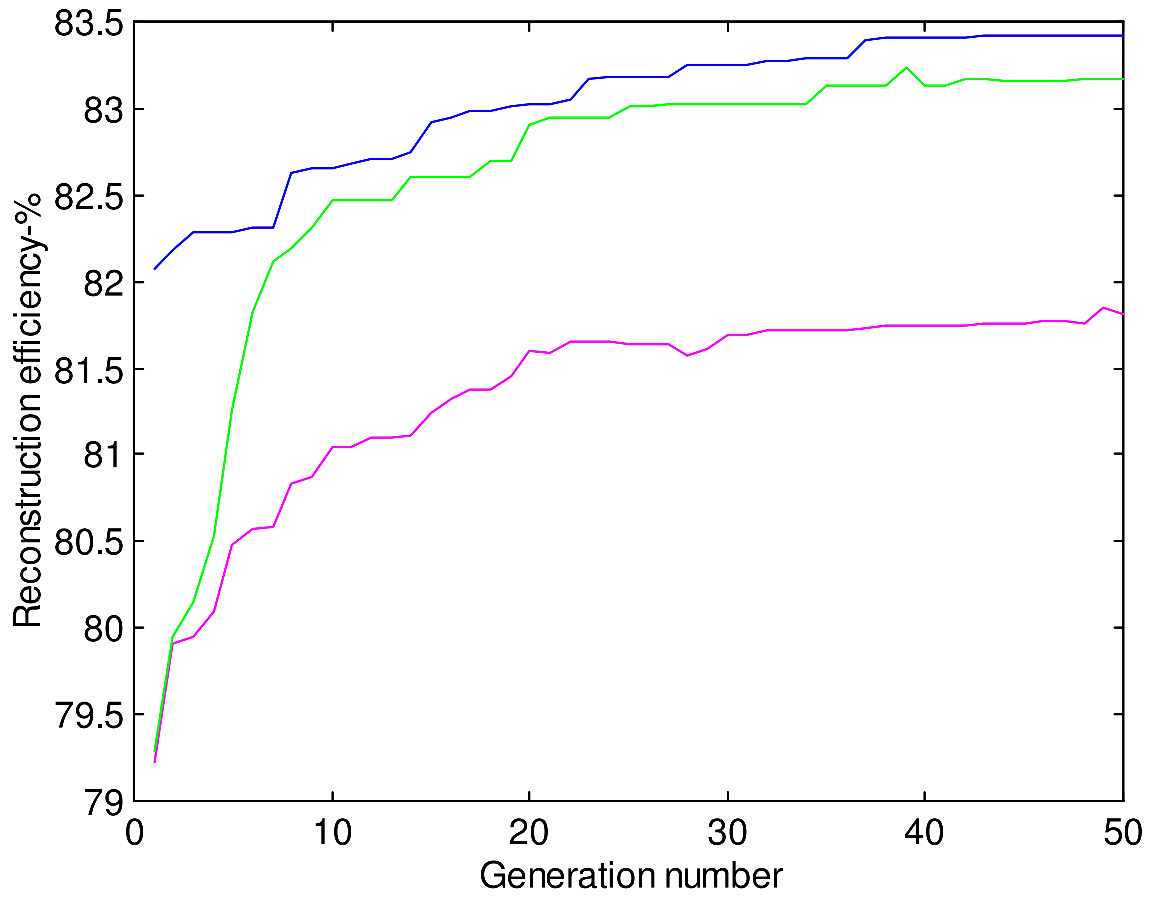
Technical details

- **Program is written in C++, compiles with any flavor of MPI (or even without it when only GFAL is used)**
- **Specific communication interfaces are implemented as subclasses of two abstract types: execution queue (for master-slave interactions) and migrator (for inter-deme interactions). This makes adding new interfaces easy.**
- **User can choose real or binary genome (the latter one can be NCB or Gray-coded)**
- **Program can be either linked with optimized application, or treat it as an external entity (just generate parameters file and call provided command)**
  - We use the second option, because overhead on that is marginal, and we gain flexibility
- **Multiple runs (up to the validity of the proxy)**
- **Many other options ...**

Results

- **Program scales very well (at least with MPI local execution queue)**
  - GFAL variant at 200 CPUs has about 25 seconds delay on one individual (in MPI variant it is order of milliseconds)
- **Migrations (GFAL+RFIO) introduce 0.3s delay if they are not too frequent (one migration per ‘generation’, couple of demes). With higher migration rates it depends on the performance of the file server.**

5 runs, popsize 22, 50 gen



Single deme (standalone population)

Multiple demes (~ 200 CPUs), about 7 migrants per run

One big population (~ 200 individuals)

More information

- **More details about this application can be found at http://wiki.polgrid.pl/index.php/DEAG**
  - Sources will be available also from there
  - Contact to author: apadee [at] icm.edu.pl
- **More information about COMPASS experiment: http://wwwcompass.cern.ch/**
- **CORAL (Compass reconstruction and analysis) - the program we were optimizing in our test case: http://coral.web.cern.ch/coral/**