Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Department of Information Technology
and Electrical Engineering

Numerical Optimization for Parallel Computing

Christian Hafner

Institute of Electromagnetic Fields (IEF)

General structure of generation based optimizers

Loop over Ni individuals

  Initialize individual (randomly)

  Evaluate fitness (time consuming)

End initialization loop

Loop over Ng generations (PSO: Ng time steps)

  Loop over Ni individuals

    Create/modify individuals (smart)

    Evaluate fitness (time consuming)

  End inner loop

End outer loop

Parallelization of generation based optimizers

Parallelize loops over Ni individuals!

Seems to be trivial if Ni=multiple of Np (Np=number of processors)

BUT:

- Optimal value of Ni depends on problem
- Reasonable Ni may be smaller than Np
- Processors with unequal speed or load
- Fitness evaluation time may strongly depend on the properties of the individual (examples follow!)

Generation-based is not well-adapted to parallelization!

Populations in nature are often also not generation-based!

Generation-free optimization

- Optimization process should continuously generate individuals based on the knowledge available
- Never wait for knowledge currently being gained (running fitness evaluations)
- Population size may be dynamic but can be static as soon as the desired size is obtained
- Good individuals may survive for ever (why not?)
- Selection mechanisms may be the same as for generation-based optimizers
- Stop after Ne fitness evaluations (Ne=Ng*Ni)

Generation-free evolutionary strategy (ES)

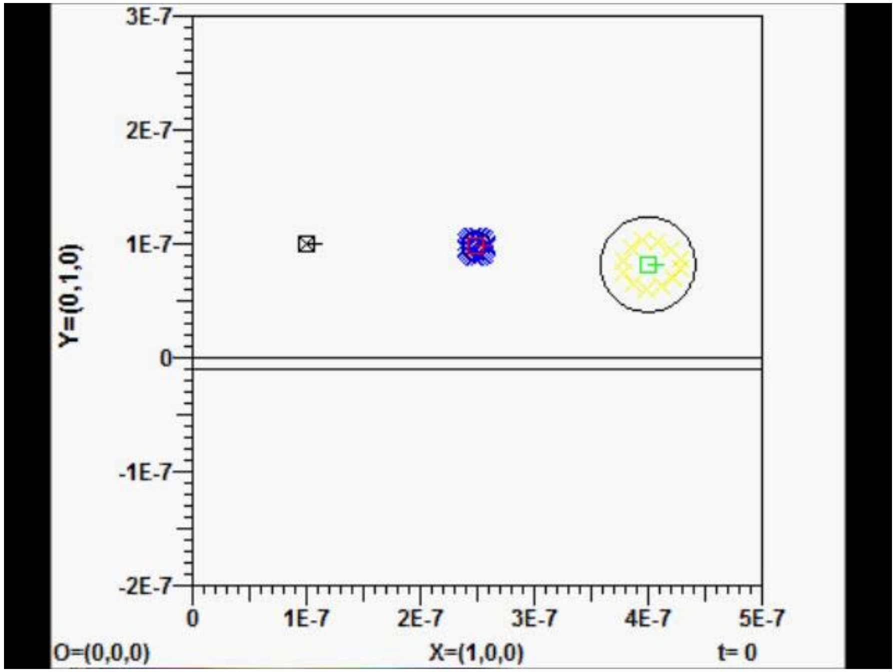
While some processor is free and N<Ne: generate individual and submit it to a free processor

As soon as a fitness has been evaluated: Insert data in table T and sort individuals in T according to fitness)

Generate individual:

- Create randomly if T shorter than Nmin
- Otherwise:
  - select 2 parents out of T (prefer good ones, different possibilities)
  - Apply crossover and mutate (as for generation-based ES)

Example: Excitation of surface wave by 2 particles I

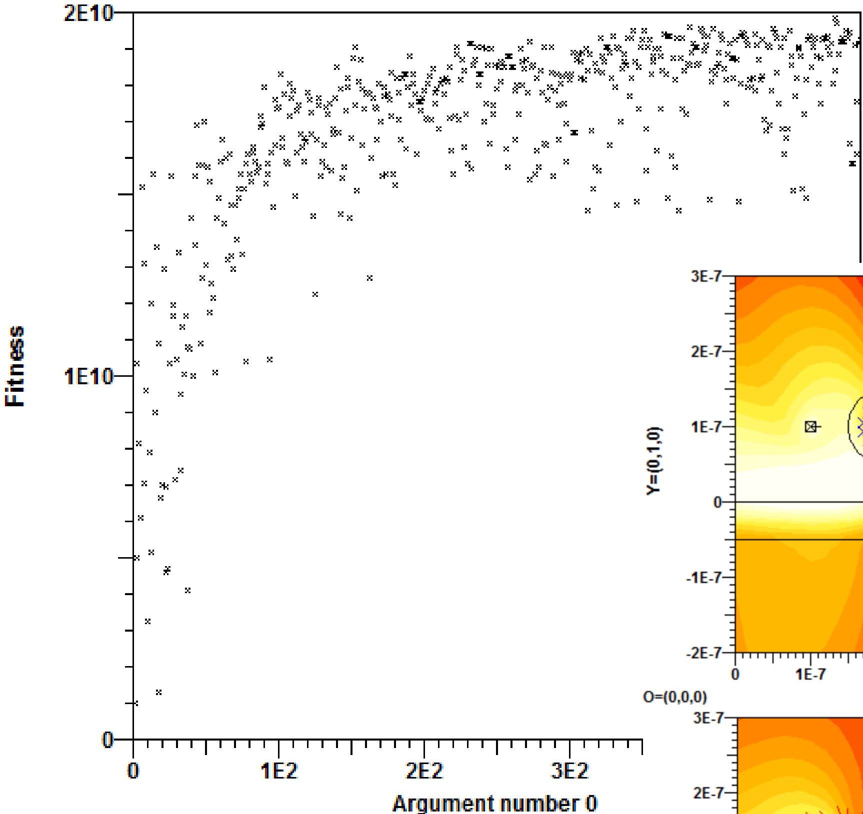


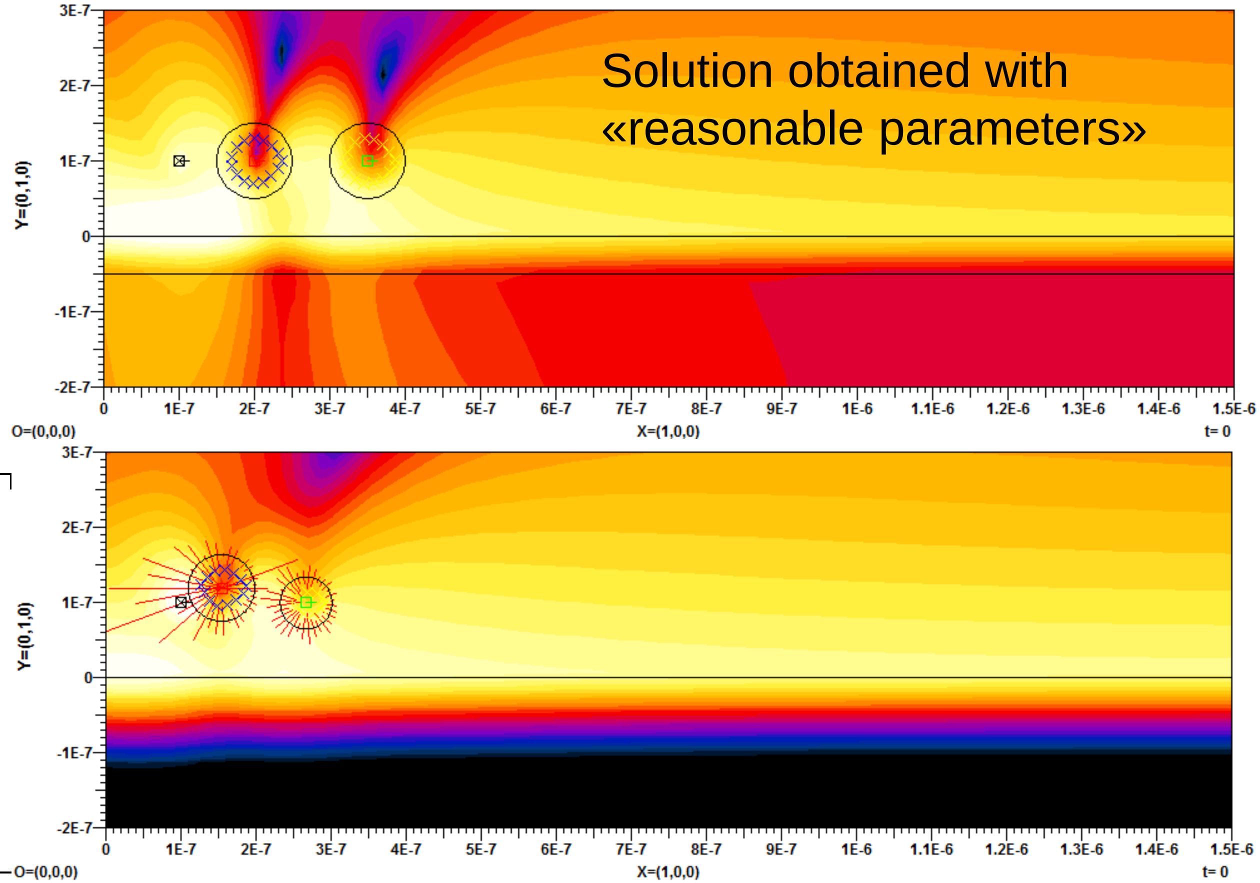
OpenMaX-MMP model by Aytac Alparslan

- Arrange 2 circular 2D particles near a point source above a metal slab in such a way that strong plasminic wave is obtained along the slab
- Vary 7 parameters: radii, distances, slab width
- Number of MMP expansions depends on particle sizes, gaps...
- Fitness evaluation times vary strongly (>factor 10)

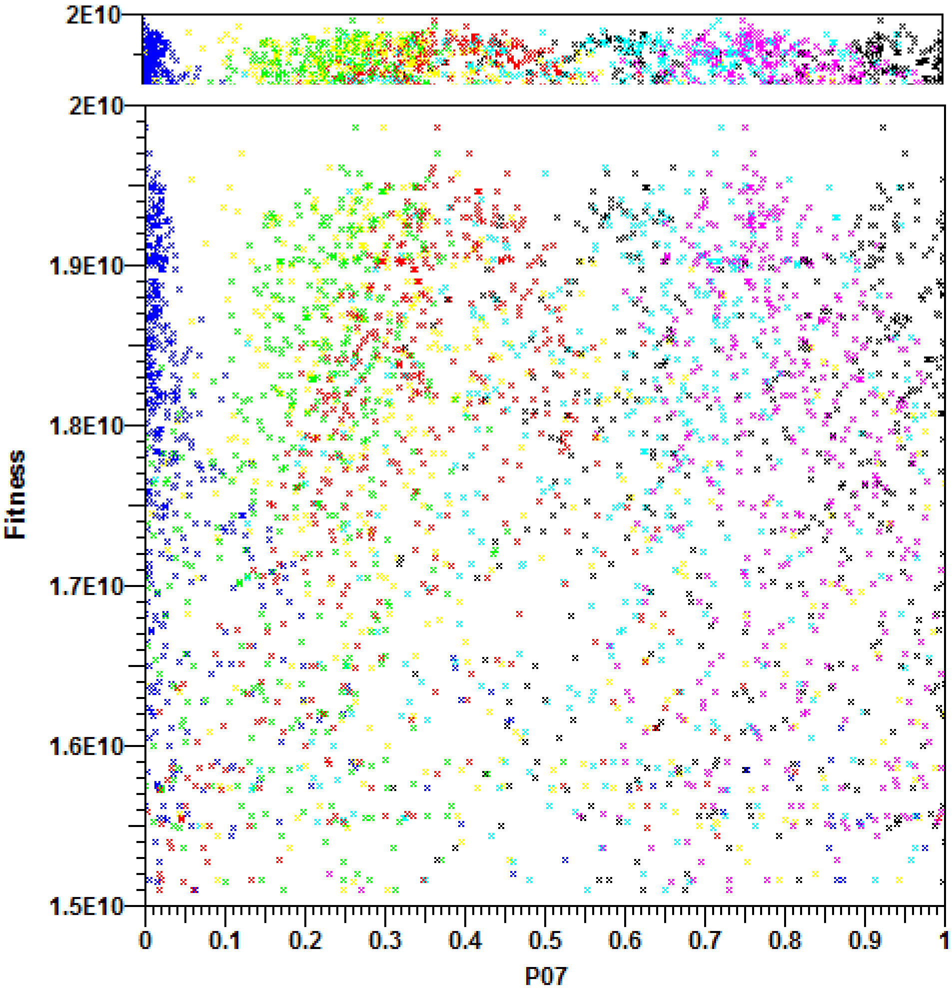
Example: Excitation of surface wave by 2 particles II

- Parallel ES performs very well
- Good solution found already after <200 fitness evaluations





Example: Excitation of surface wave by 2 particles III



Analysis of optimization results

- Has global optimum been found?
- Is best solution good enough?
- Should a longer optimization run be started with what optimization parameters?
- Was the selected parameter range reasonable?

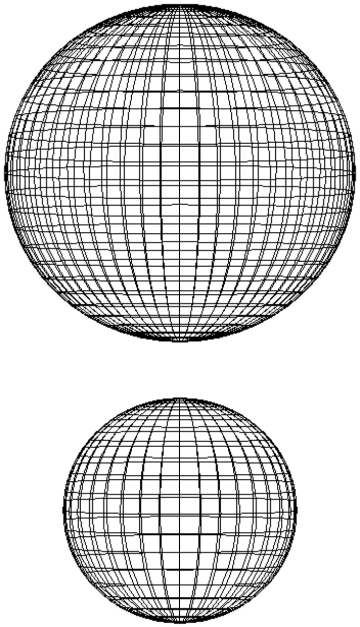
Plot Fitness versus normalized parameters (p0...p6: black, red, green, blue, yellow, pink, light blue)

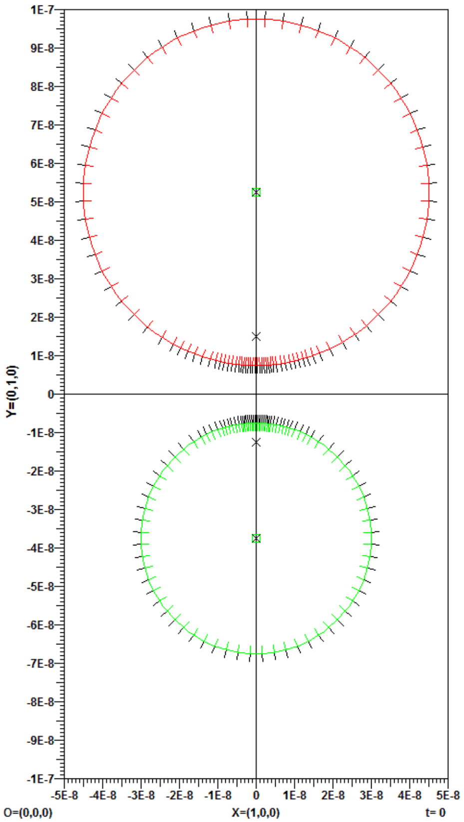
Optimal blue: 0 might be <1

Optimal black: near 1, might be >1

→ Adjust at least ranges of p0, p3

Plasmonic optical antenna I





Pair of spheres, Ag, Au

- Optimize radii and gap
- Only 3 parameters
- Full 3D model
- Fitness evluation expensive
- Expansions must be adapted to radii and gap size for saving computation time
- Computation time depends on parameters

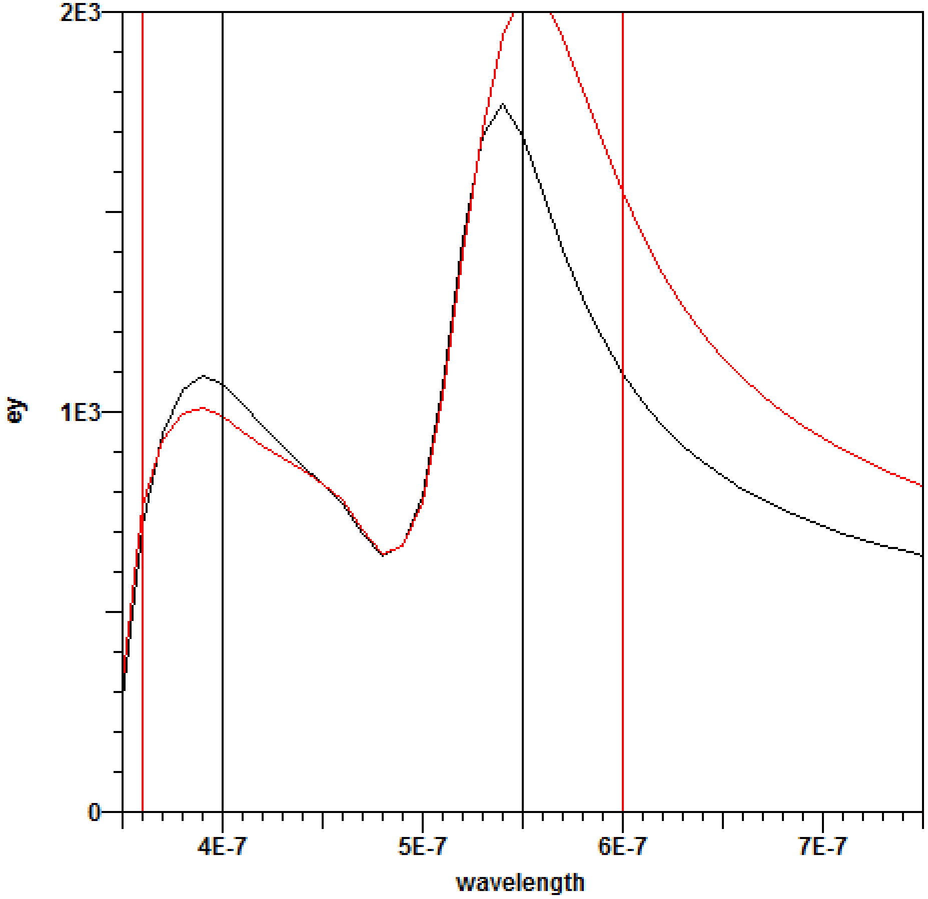
Plasmonic optical antenna II

```
!set var v0 30e-9 ! radius r1
!set var v1 30e-9 ! radius r2
!set var v2 5e-9 ! gap g
! all other variables computed using formula
set var v3 for add(v0,div(v2,2)) ! center y1
set var v4 for neg(add(v1,div(v2,2))) ! center y2
set var v5 for div(mul(sub(v3,v4,),v0),add(v0,v1)) ! d1
set var v6 for div(mul(sub(v3,v4,),v1),add(v0,v1)) ! d2
set var v7 for sub(v3,div(sqr(v0),v5)) ! y1'
set var v8 for add(v4,div(sqr(v1),v6)) ! y2'
! set boundaries (circles)
set bou 1 cor 1 0 v3 v0
set bou 2 cor 1 0 v4 v1
!dra bou 0
! set 3d expansions
set exp 1 3dl 0 v3 0
set exp 2 3dl 0 v4 0
set exp 3 3dl 0 v8 0
set exp 4 3dl 0 v7 0
set exp 5 3dl 0 v3 0
set exp 6 3dl 0 v4 0
set exp 7 3dl 0 v7 0
set exp 8 3dl 0 v8 0
! set auxiliary 2d multipoles for matching point setting
set exp 9 loc 0 v3
set exp 10 loc 0 v4
set exp 11 loc 0 v7
set exp 12 loc 0 v8
```

Adapting 3D MMP model

- Optimizer provides 3 parameters v0, v1, v2
- Skript language
- Evaluate auxiliary variables v3...v8 using formula interpreter
- Compute locations of 3D multipole expansions
- Set auxiliary 2D multipoles for adapting the matching points

Plasmonic optical antenna III

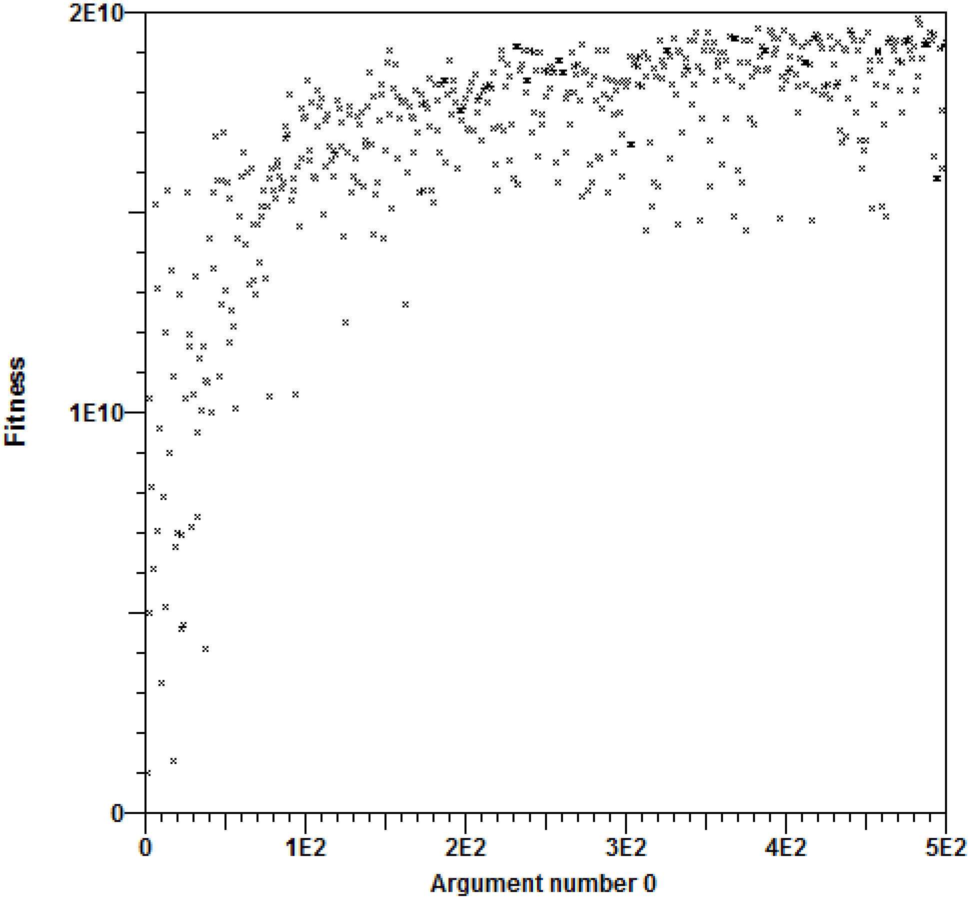


Different goals possible

e.g. maximum response at two wavelengths l1,l2

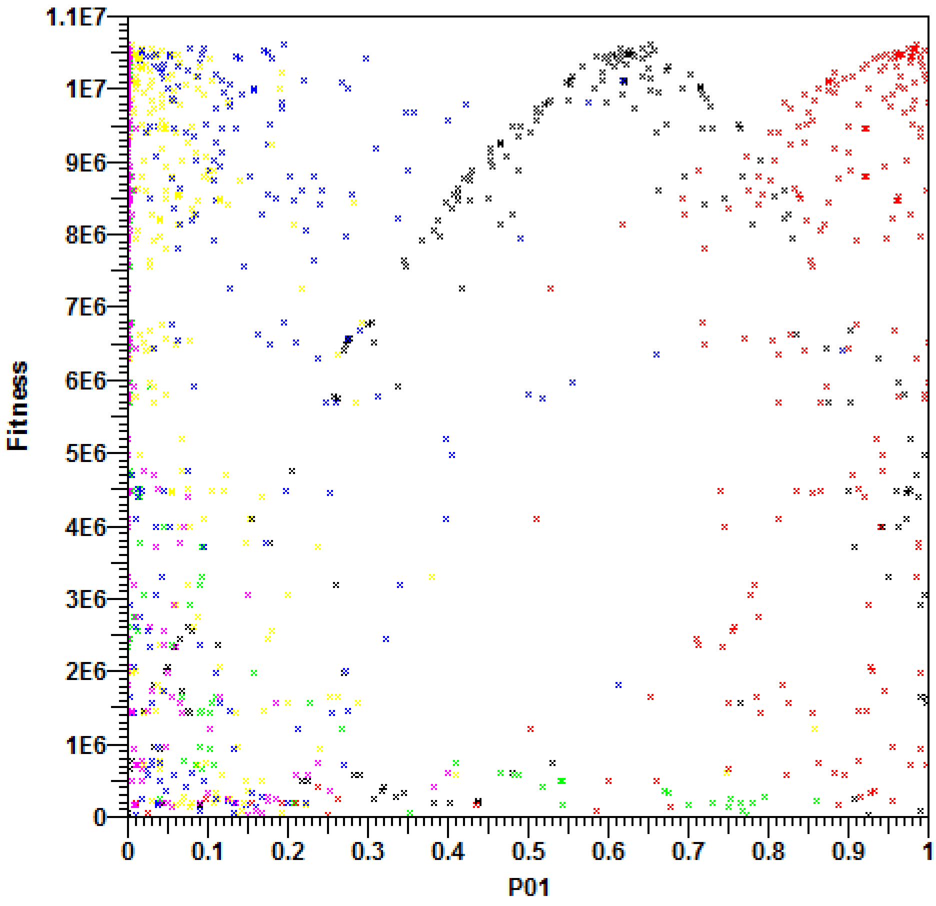
- Solution depends on l1,l2
- High quality solutions only for certain selections of l1, l2
- Better solutions: shape optimization, other materials,
  → more parameters

Plasmonic optical antenna IV



- Good fitness obtained very quickly
- Good results after < 200 fitness evaluations
- Parallel ES performs very well

Plasmonic optical antenna V



Check parameter range

- Black: R1 OK, might be reduced
- Red: R2 should be extended to larger values
- Green: Gap should be as narrow as posible!

Check sigma values

- All sigma reduced with increasing fitness
- Blue: sigma(R2) keeps highest values
- Pink: sigma(Gap) tends to very small values