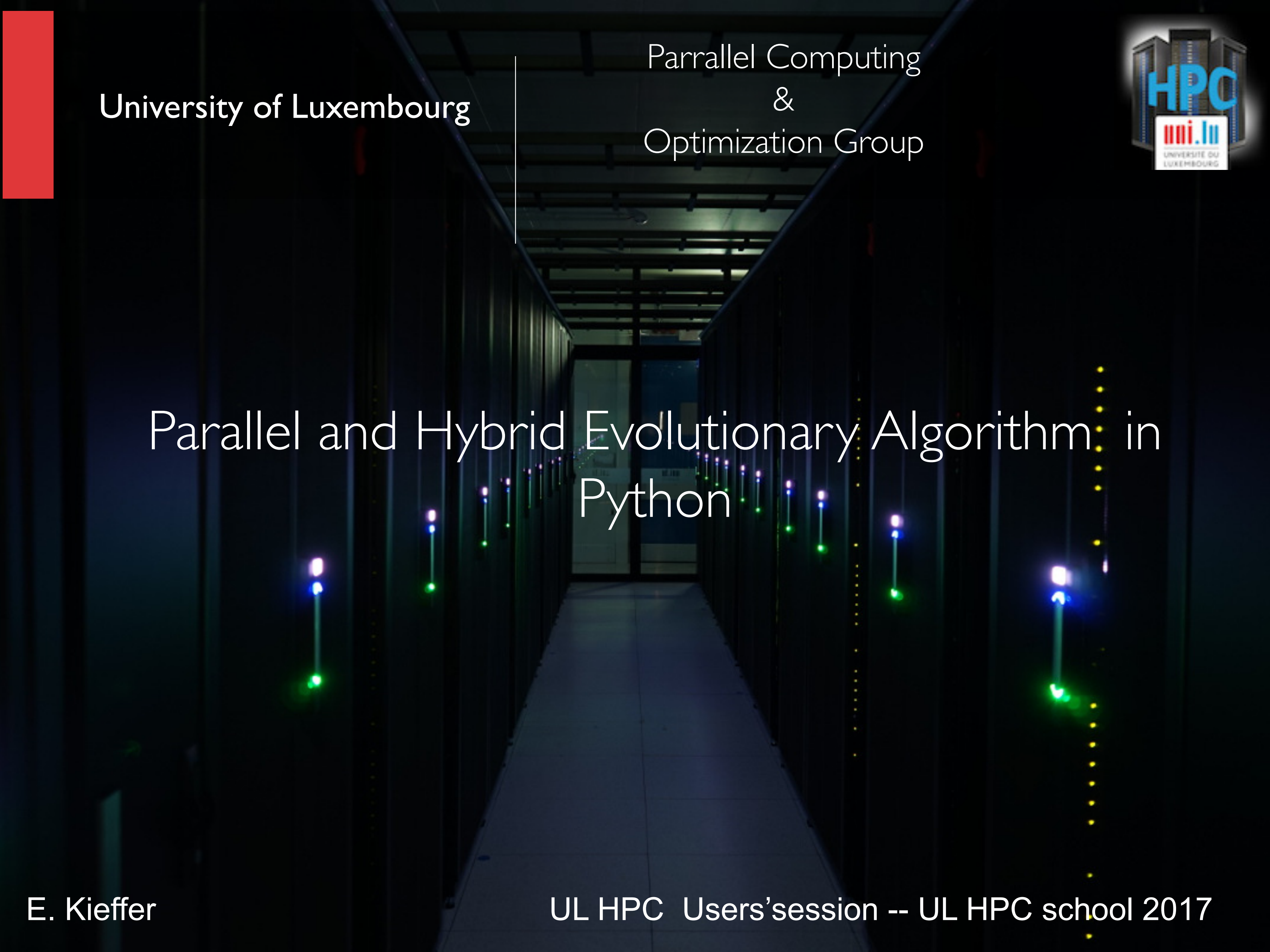


University of Luxembourg

Parrallel Computing
&
Optimization Group

Parallel and Hybrid Evolutionary Algorithm in Python

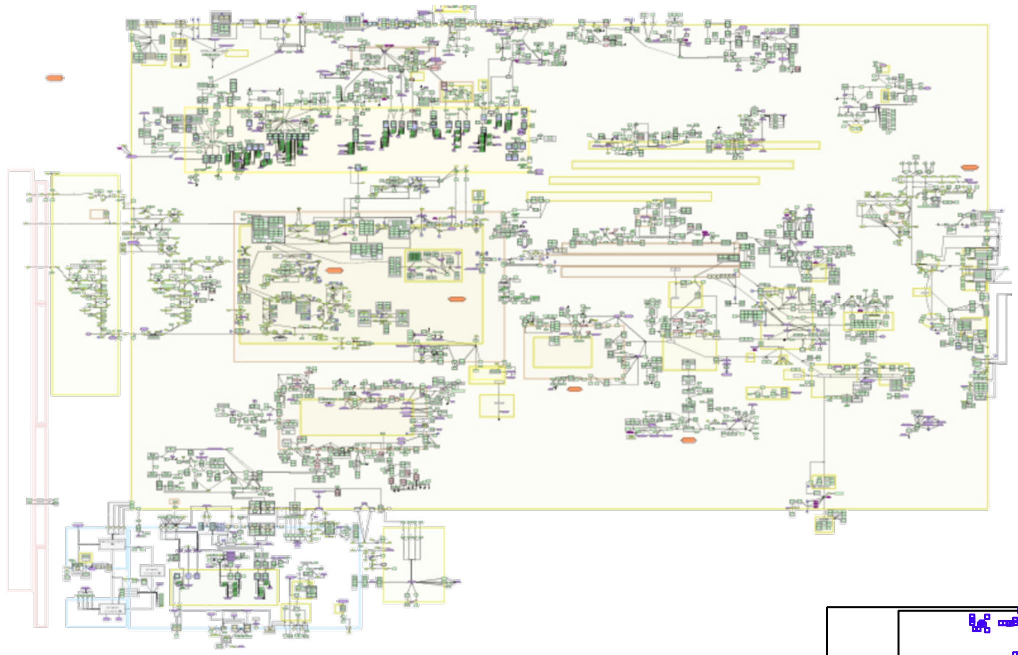
E. Kieffer

UL HPC Users'session -- UL HPC school 2017

Contents

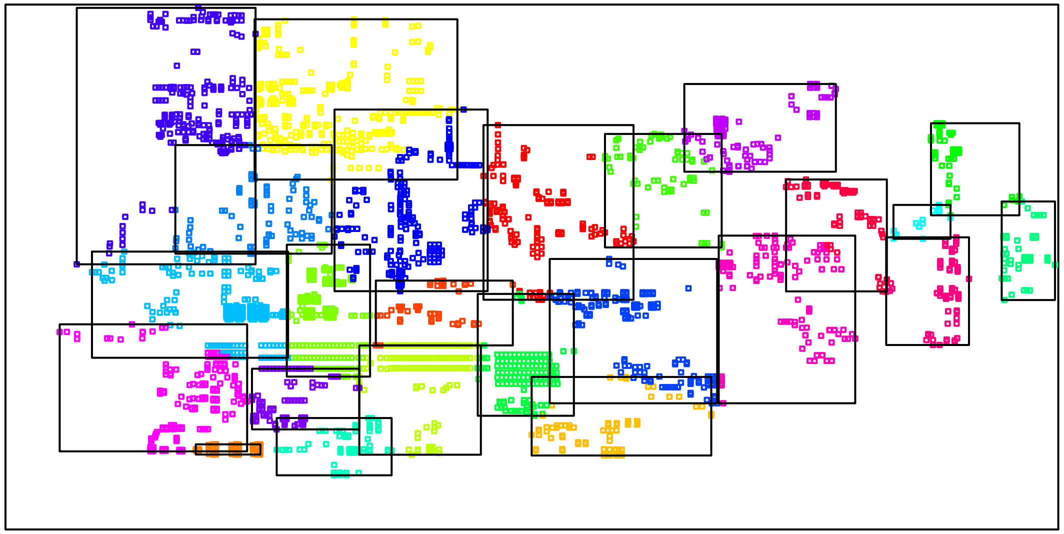
CONTEXT & MOTIVATION

Parkinson Disease Map



- Large (hyper-)Graph
- Extract Knowledge
- First experiments with standard Clustering approach

- Hierarchical Clustering
- Several metric (e.g. GO, NET, EU)
- Hard to combine



Bi-level Clustering

- Clustering often based on a two phase algorithm:
  - Find cluster representatives
  - Assign data to clusters
- Generally the same metric is used for both steps
- Consider these two steps as two nested optimization problems with different metrics
- Metric:
  - Euclidean distance
  - Network distance
  - Distance based on Gene/Disease Ontology
- Use Evolutionary Algorithm (EA) to solve the Bi-level Clustering problem
- Use MOEA to detect the number of clusters

Bi-level Optimization

- Bi-levels ←→ Nested problems

- A problem constraining another one → NP-hard even for convex levels

Upper-level

$$\begin{aligned} \min \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \end{aligned}$$

$$\begin{aligned} \min \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \leq 0 \\ & x, y \geq 0 \end{aligned}$$

Lower-level

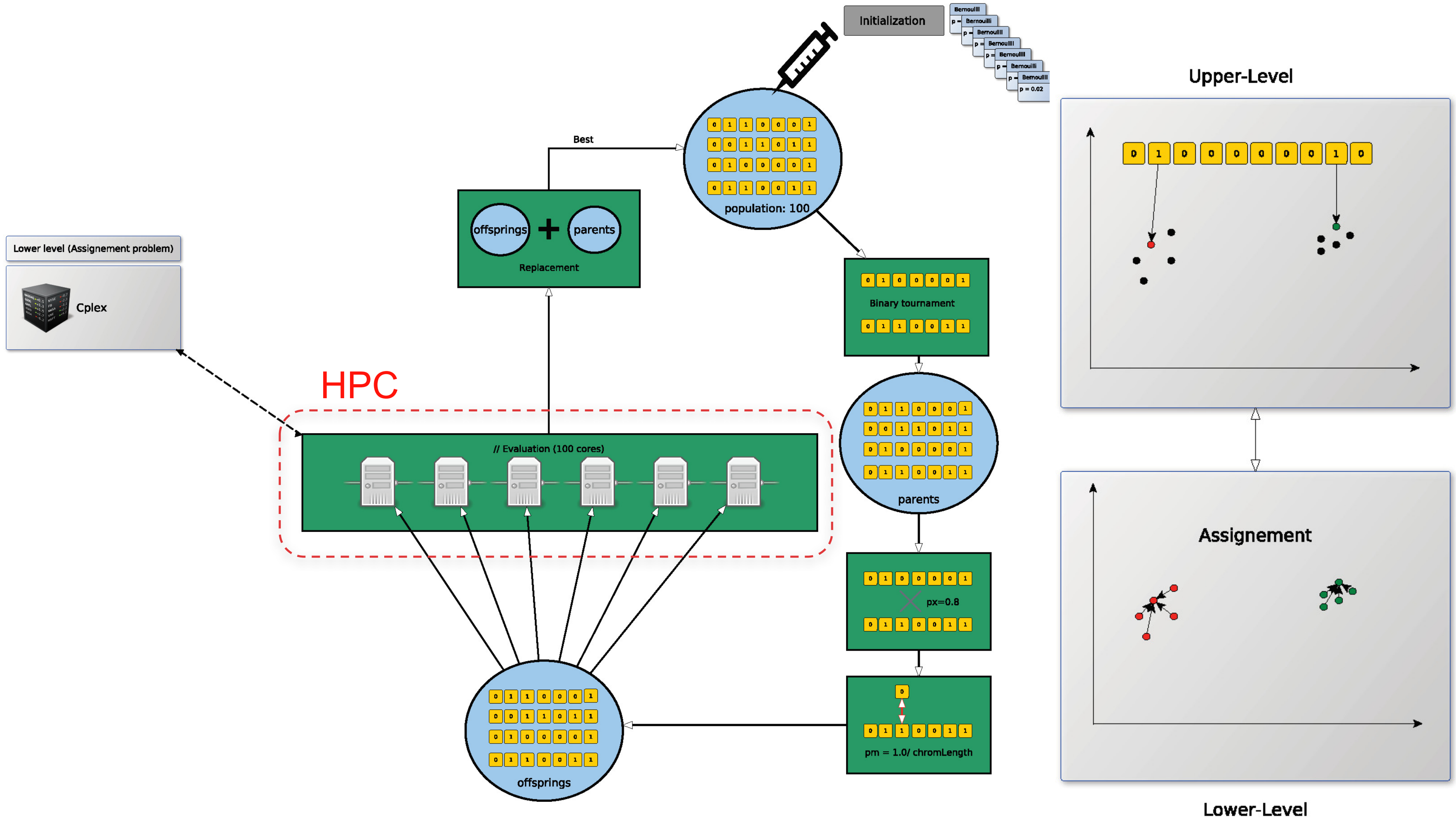
Bi-level Clustering

d^1_{ij} and d^2_{ij} are respectively the distances considered for the first and second level. The decision variables are:

- $Y_j = \begin{cases} 1 & \text{if point j become a centroid} \\ 0 & \text{else.} \end{cases}$
- $X_{ij} = \begin{cases} 1 & \text{if point i belongs to cluster j} \\ 0 & \text{else.} \end{cases}$

$$\begin{aligned}
\min \quad & F = \sum_i \sum_j (d^1_{ij} X_{ij}, \sum_j Y_j) \\
\text{s.t.} \quad & \min f = \sum_i \sum_j d^2_{ij} X_{ij} \\
& \text{s.t.} \ \sum_j X_{ij} = 1 \quad \forall j \in \{1, ..., \sum_j Y_j\} \\
& \qquad X_{ij} - Y_j \leq 0 \quad \forall j \in \{1, ..., \sum_j Y_j\} \\
& \qquad X_{ij}, Y_j \in \{0, 1\}
\end{aligned}$$

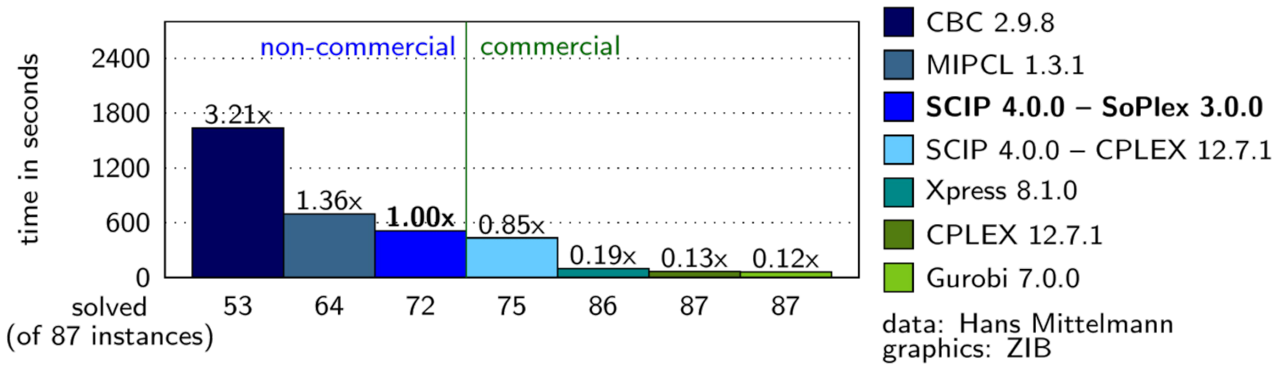
Parallel and hybrid EA



PYTHON TOOLS ON THE UL HPC PLATFORM

Using CPLEX on the UL HPC

- IBM ILOG CPLEX Optimizer's mathematical programming technology.
- One of the most efficient solver on the market:



- CPLEX available for HPC user with IBM Academic Initiative membership
  - Need first to register to the IBM Academic Initiative:
    - https://developer.ibm.com/academic/
  - Forward the membership confirmation mail to the HPC admins
- To use CPLEX on the cluster:

  -
    ```
    $ module use $PROJECTWORK/cplex/soft/modules
    $ module load CPLEX
    ```

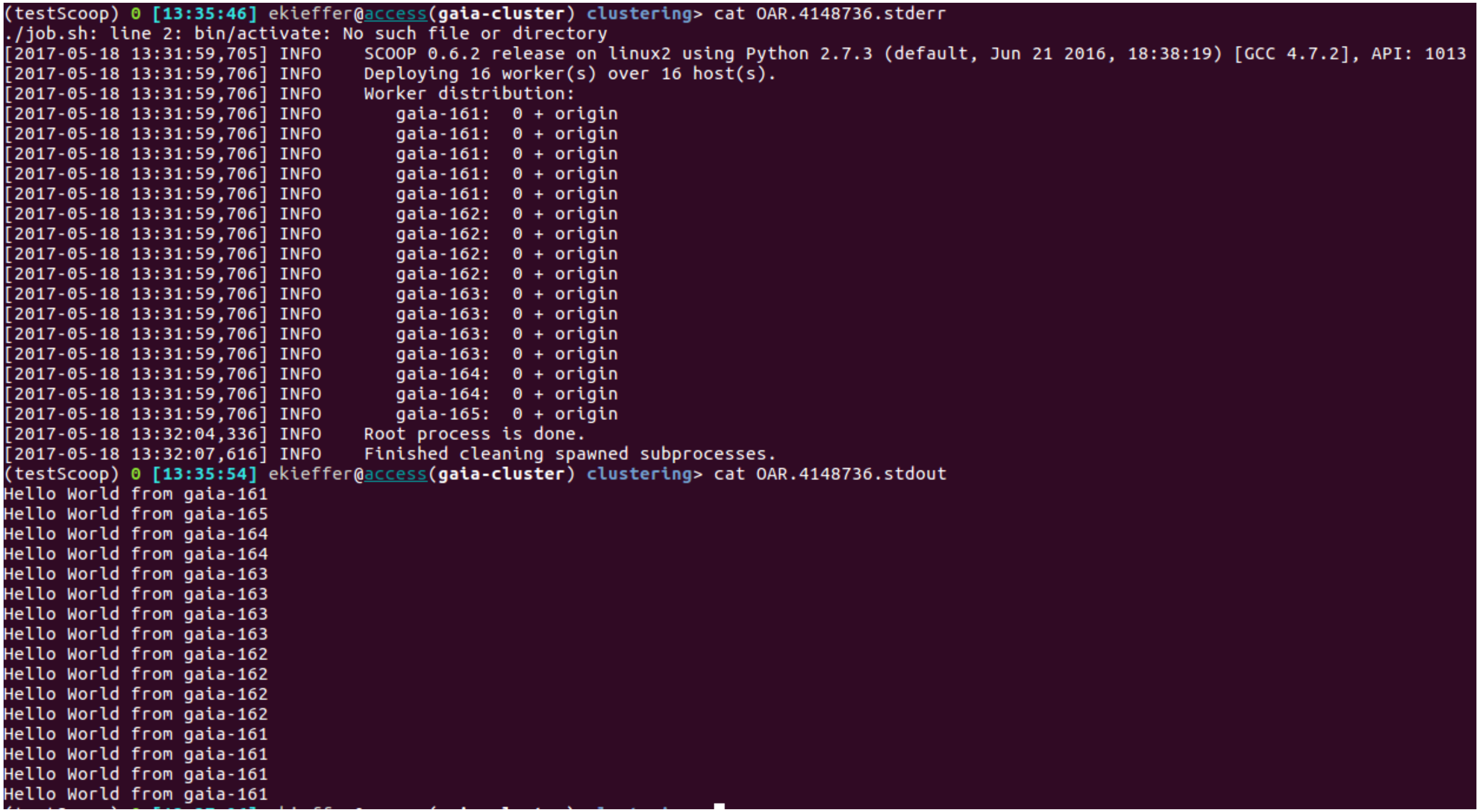
Parallel Evaluations with SCOOP

- Scalable COncurrent Operations in Python
  - is a distributed task module
  - concurrent parallel programming
  - on various environments, from heterogeneous grids to supercomputers
- Command to execute a python script using SCOOP
  - *python -m scoop --hostfile $OAR_NODEFILE -n 16 --ssh-executable "oarsh" hello.py*
- Parameters:
  - --hostfile: path to the file contains all hostnames
  - --ssh-executable: the command to access nodes (here oarsh)
  - -n: the number of workers

Hello.py →

```python
from __future__ import print_function
from scoop import futures
import socket
def helloWorld(value):
    return "Hello World from{0}".format(socket.gethostname())
if __name__ == "__main__":
    returnValues = list(futures.map(helloWorld, range(16)))
    print("\n".join(returnValues))
```

Example

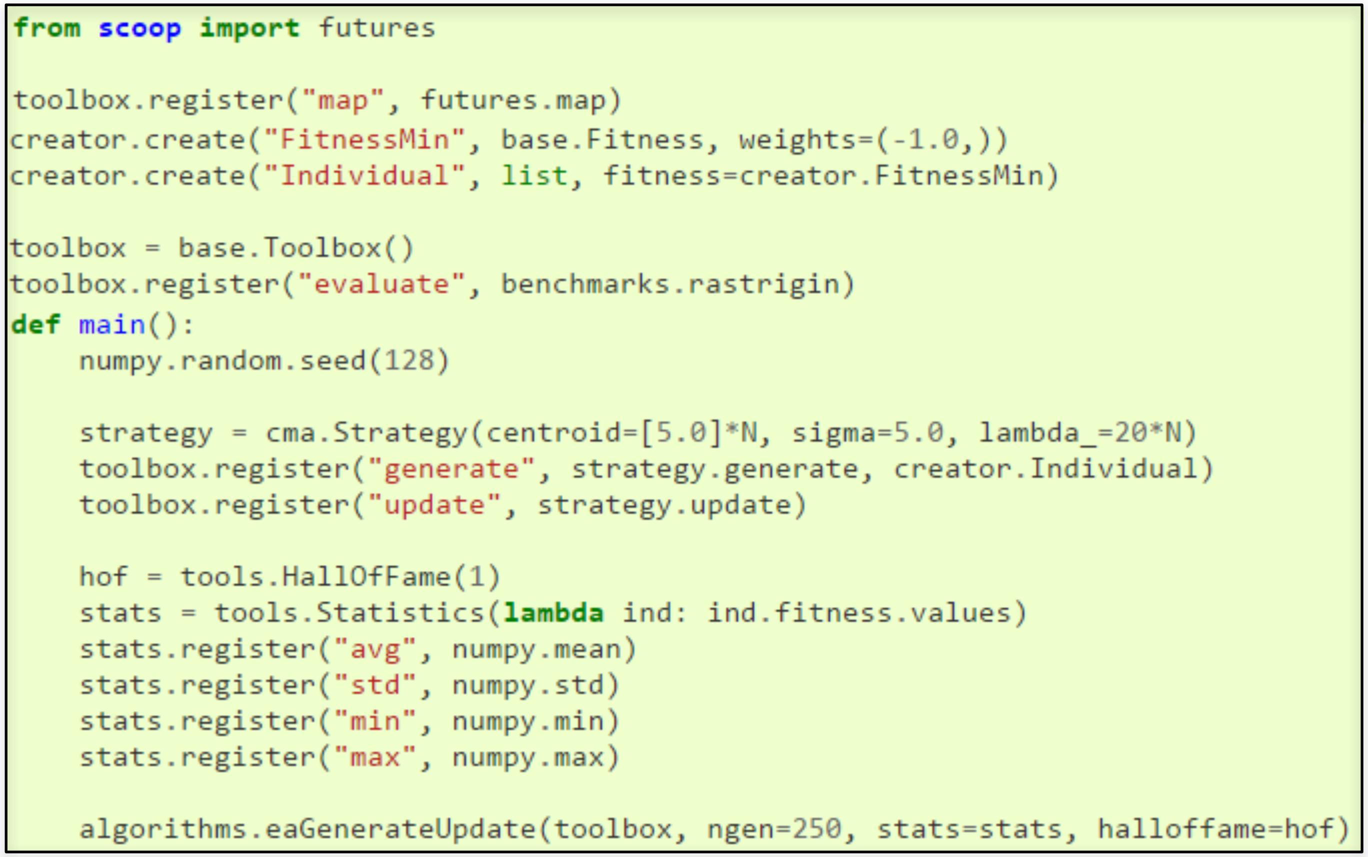


```
(testScoop) 0 [13:35:46] ekieffer@access(gaia-cluster) clustering> cat OAR.4148736.stderr
./job.sh: line 2: bin/activate: No such file or directory
[2017-05-18 13:31:59,705] INFO    SCOOP 0.6.2 release on linux2 using Python 2.7.3 (default, Jun 21 2016, 18:38:19) [GCC 4.7.2], API: 1013
[2017-05-18 13:31:59,706] INFO    Deploying 16 worker(s) over 16 host(s).
[2017-05-18 13:31:59,706] INFO    Worker distribution:
[2017-05-18 13:31:59,706] INFO       gaia-161:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-161:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-161:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-161:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-161:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-162:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-162:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-162:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-162:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-163:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-163:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-163:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-163:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-164:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-164:  0 + origin
[2017-05-18 13:31:59,706] INFO       gaia-165:  0 + origin
[2017-05-18 13:32:04,336] INFO    Root process is done.
[2017-05-18 13:32:07,616] INFO    Finished cleaning spawned subprocesses.
(testScoop) 0 [13:35:54] ekieffer@access(gaia-cluster) clustering> cat OAR.4148736.stdout
Hello World from gaia-161
Hello World from gaia-165
Hello World from gaia-164
Hello World from gaia-164
Hello World from gaia-163
Hello World from gaia-163
Hello World from gaia-163
Hello World from gaia-163
Hello World from gaia-162
Hello World from gaia-162
Hello World from gaia-162
Hello World from gaia-162
Hello World from gaia-161
Hello World from gaia-161
Hello World from gaia-161
Hello World from gaia-161
```

DEAP library for Evolutionary Computation in Python

- https://github.com/DEAP/deap
- Rapid prototyping and testing of ideas
- Parallelization mechanism based on SCOOP
- CMA-ES algorithm

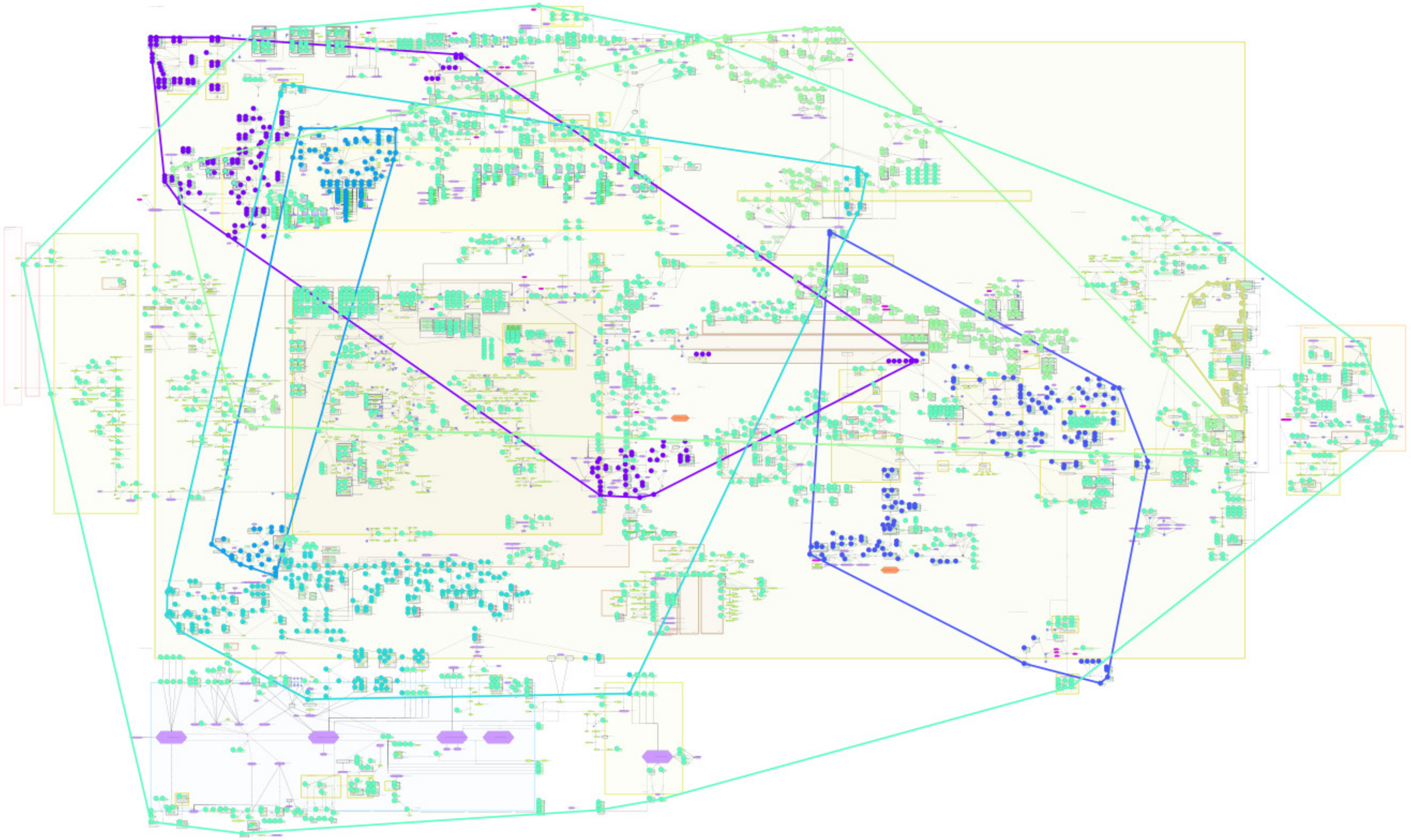
DISTRIBUTED EVOLUTIONARY ALGORITHMS IN PYTHON



```python
from scoop import futures

toolbox.register("map", futures.map)
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
toolbox.register("evaluate", benchmarks.rastrigin)
def main():
    numpy.random.seed(128)

    strategy = cma.Strategy(centroid=[5.0]*N, sigma=5.0, lambda_=20*N)
    toolbox.register("generate", strategy.generate, creator.Individual)
    toolbox.register("update", strategy.update)

    hof = tools.HallOfFame(1)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", numpy.mean)
    stats.register("std", numpy.std)
    stats.register("min", numpy.min)
    stats.register("max", numpy.max)

    algorithms.eaGenerateUpdate(toolbox, ngen=250, stats=stats, halloffame=hof)
```

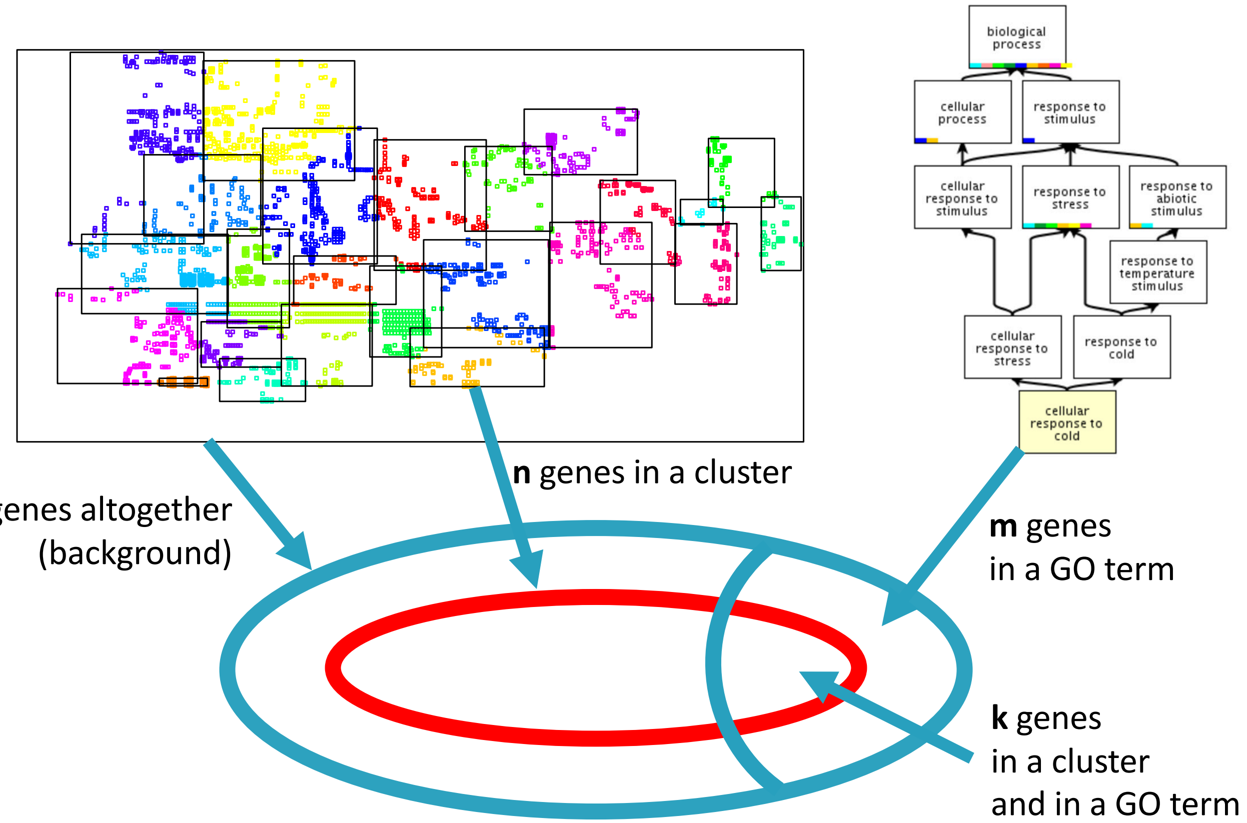
EXPERIMENTS & VALIDATION

Clustering results



Bi-level Clustering

Enrichment analysis: hypergeometric test

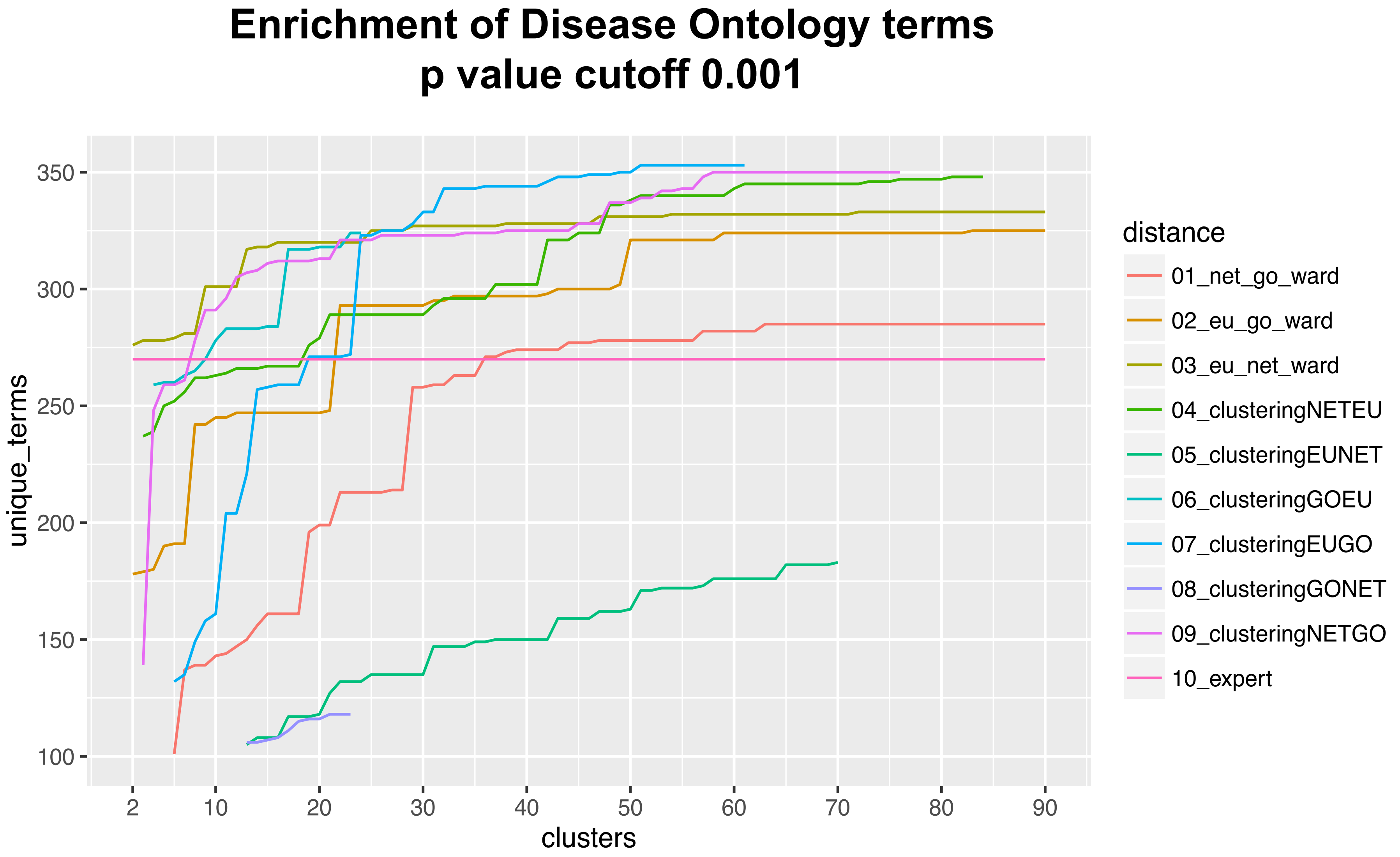

biological process
cellular process
response to stimulus
cellular response to stimulus
response to stress
response to abiotic stimulus
response to temperature stimulus
cellular response to stress
response to cold
cellular response to cold
n genes in a cluster
N genes altogether (background)
m genes in a GO term
k genes in a cluster and in a GO term

$$P(X=k)=\frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}}$$

Adapted from: Florian Markowetz
Network Biology Lent 2010

A cluster represents a sample of *n* genes from a total population of N genes. It is know that the considered GO term contains m genes. What is the probability to have the same k genes in our cluster and in the considered GO term ?

Bi-level Clustering



Conclusions

- Knowledge extraction on the Parkinson Disease MAP
- Bi-level clustering model
- Solve the model with Hybrid and Parallel EA
- Experiments required a lot of resources → UL HPC Platform
  - Hybrid → CPLEX solver
  - Parallel → SCOOP library for parallel evaluations
  - Evolutionary Computation → DEAP library

Questions ?

Thank you
for
your attention

PS9 (13h30 – 15h30):
Advanced Prototyping with python presented by Clement Parisot