PARADISEO (PARAllel and DIStributed Evolving Objects)

Sébastien CAHON[1],

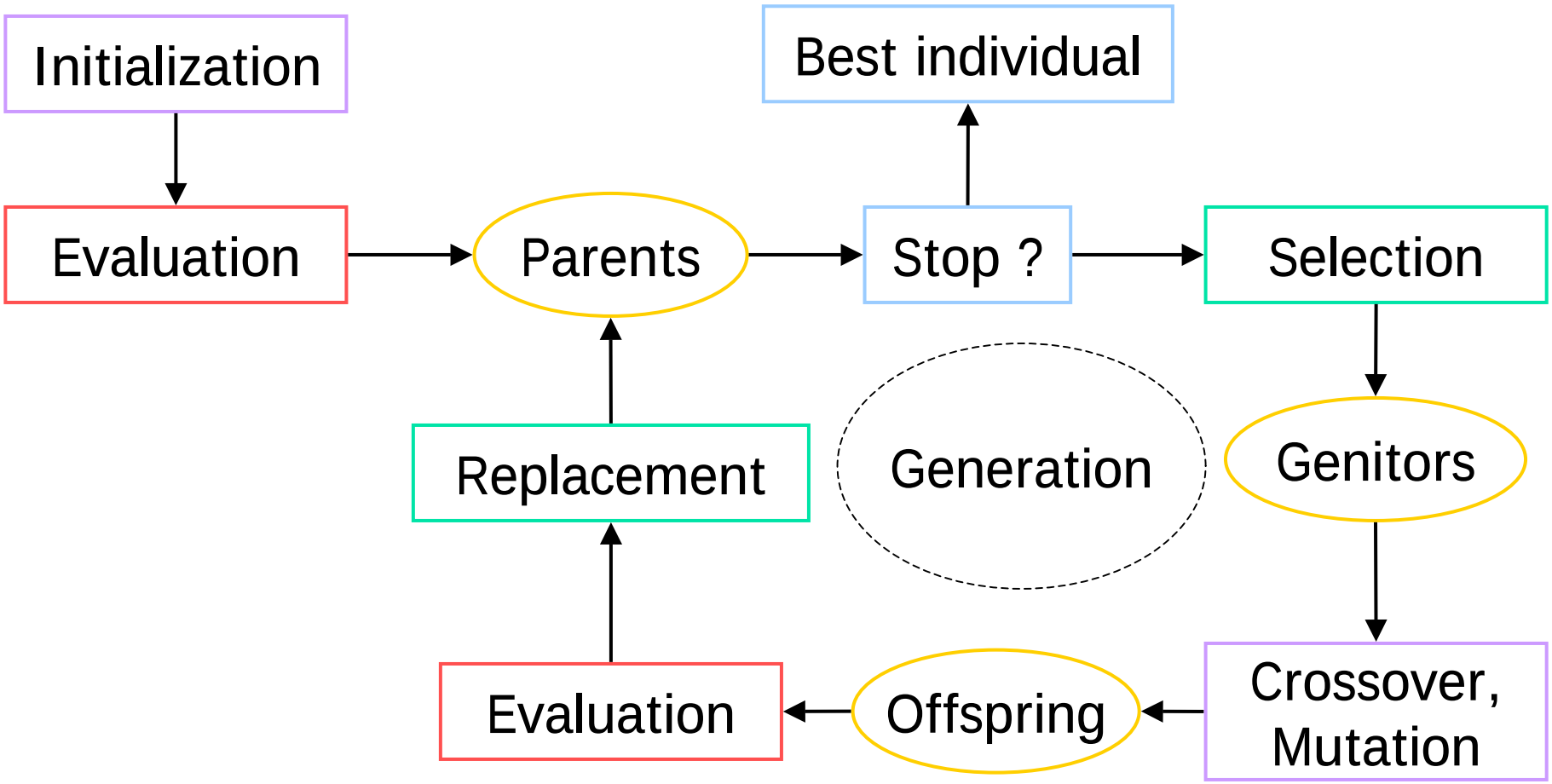
El-Ghazali TALBI[1],

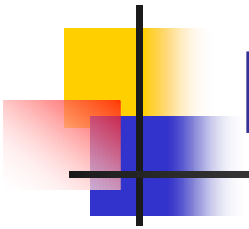
Marc SHOENAUER[2]

1. Laboratoire D'Informatique Fondamentale de Lille

2. INRIA Rocquencourt

Recall : The scheme of an evolutionary algorithm

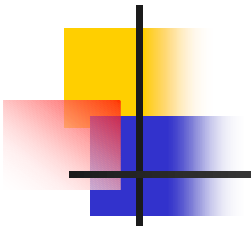




How does it work ?

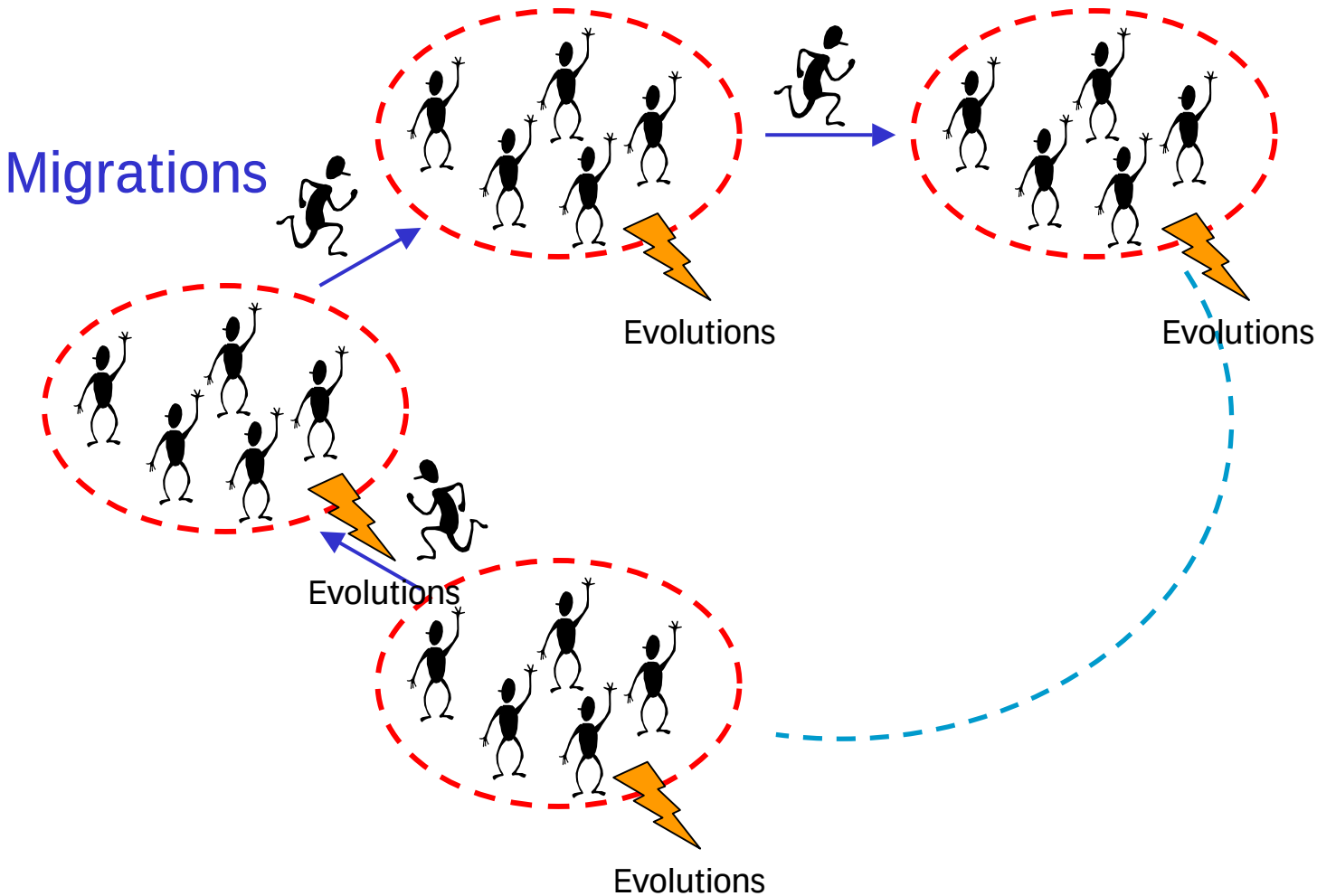
- Ideas from natural selection
  - Survival of the best-fitted individuals
  - Use of genetic operators
- Modeling
  - An environment $\rightarrow$ A Problem
  - Objects evolving in this environment $\rightarrow$ Solutions to this problem
- Convergence of the population to a set of particularly well-fitted individuals

The reasons of the parallelization of E.A.



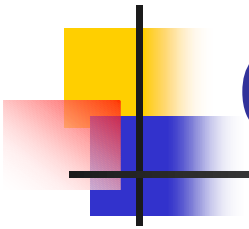
- E.A. work in a naturally parallel way
- High costs
  - CPU ressources
  - Memory
- Obtaining an improved robustness

The island model : A first example



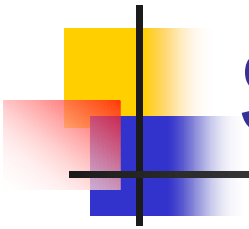
The island model

- A population distributed among a set of islands
- Local work
  - Cycle of evolutions
    - Selection
    - Breeding
    - Replacement
  - (Ir)regular migrations of solutions between these colonies



Contribution

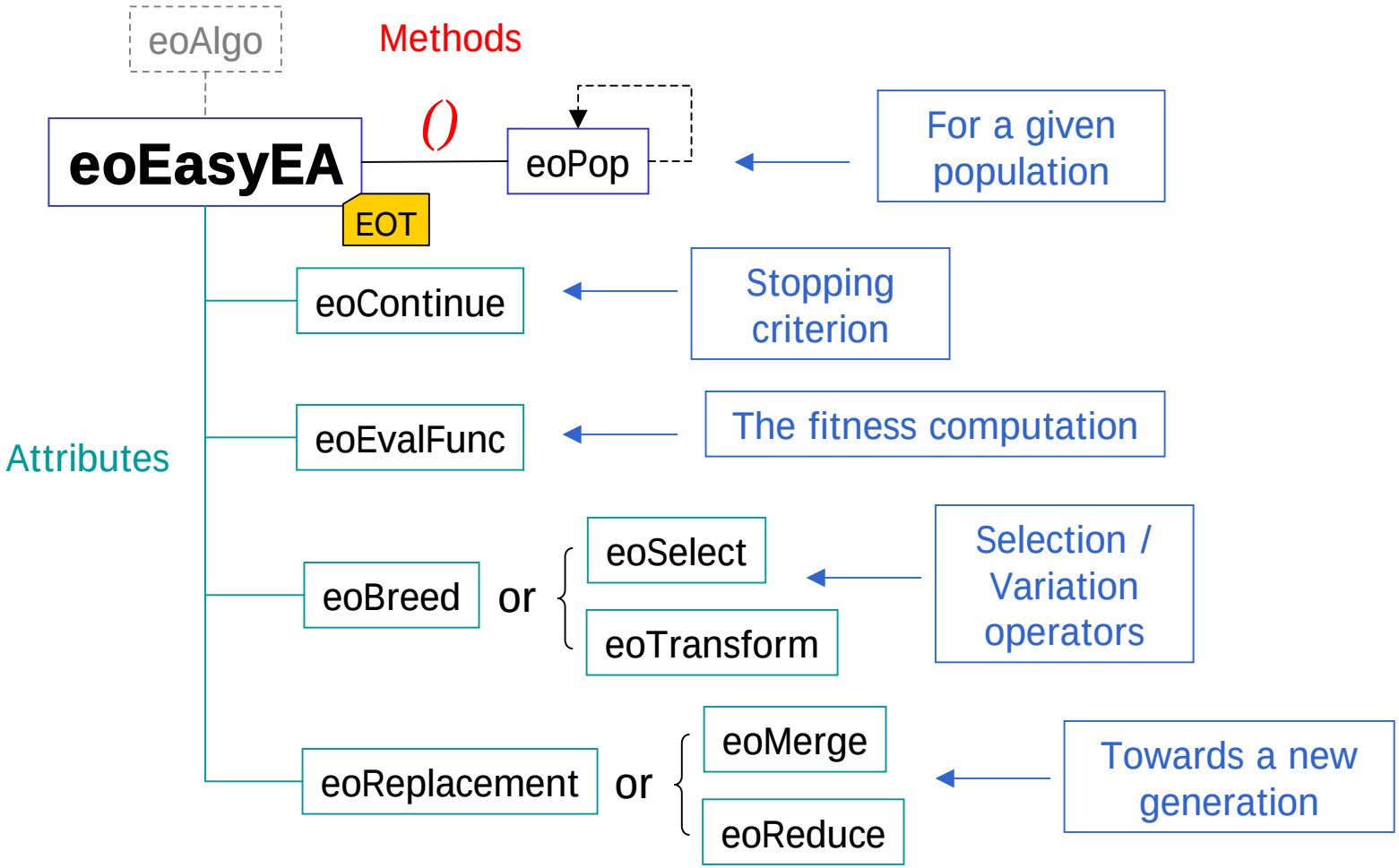
- A delayed convergence of the whole population
  - A significative improved robustness
  - Better solutions



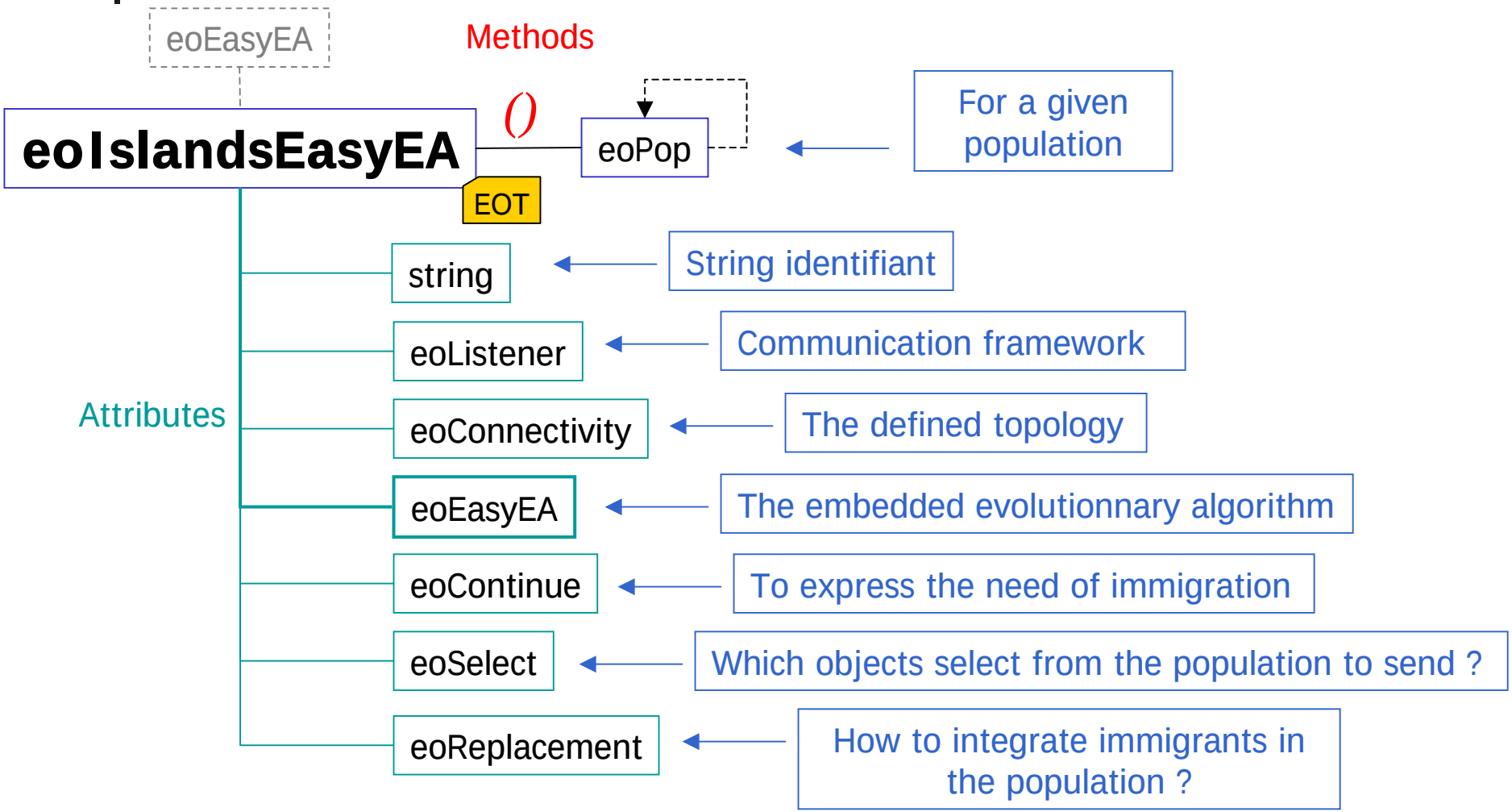
Some parameters

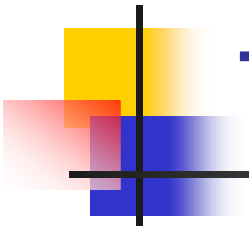
- Communication
  - Synchronous
    → Frequency of migrations
  - Asynchronous
    → Bound to an event (stagnation, ...)
- Composition of such migrations
  - Size (from 1 to 10% of the local population)
  - Selection (elitist, random, ...)
  - Replacement
- The used topology (ring, ...)

The internal structure of an evolutionary algorithm in EO



The islands model in EO



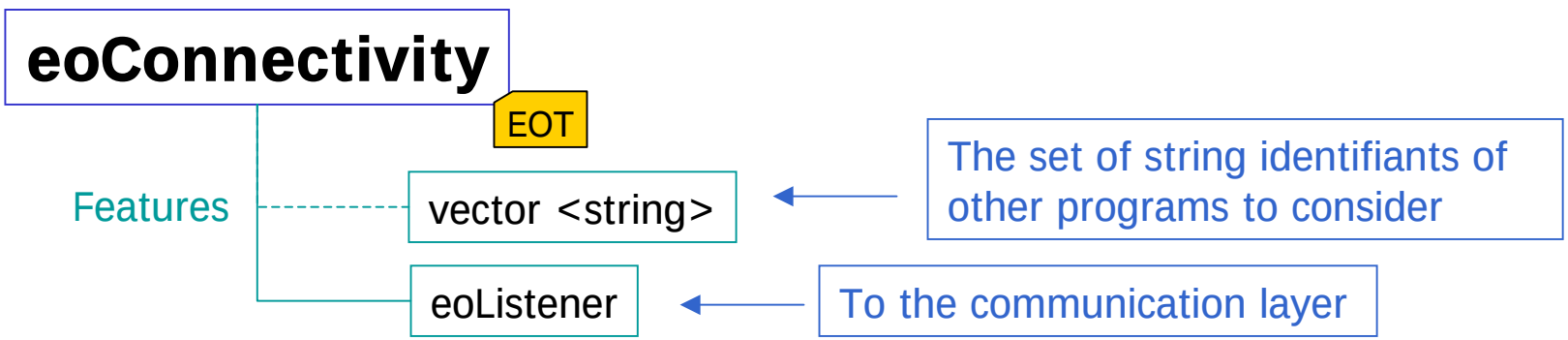


The « eoListener » component

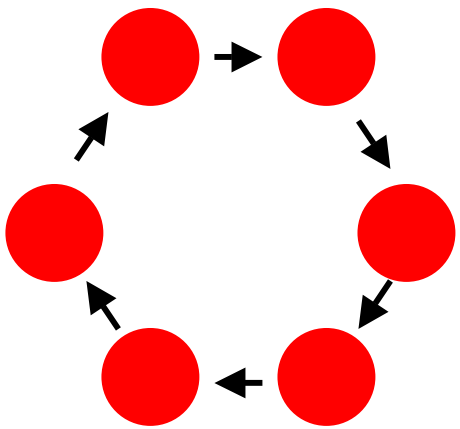
- Necessary to any parallel model
- Parameterized by the type of exchanged objects
- Enables to observe a global state of the parallel model
  → Identification of the distributed agents
- Services
  - To send/receive objects
  - To express a need of immigration
  - ...
- It must be built once in any program

The « eoConnectivity » component

- An abstract class
- Defines the graph of communication (I/O of any island)

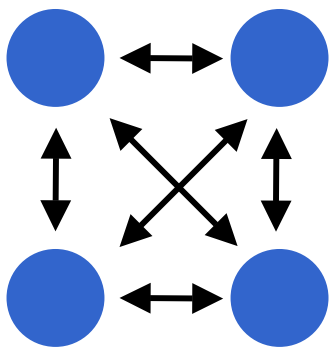


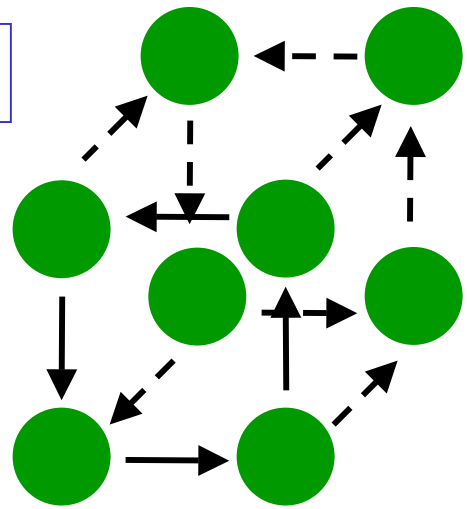
Some predefined topologies



eoRingConnectivity

eoFullConnectivity





eoHyperCubeConnectivity

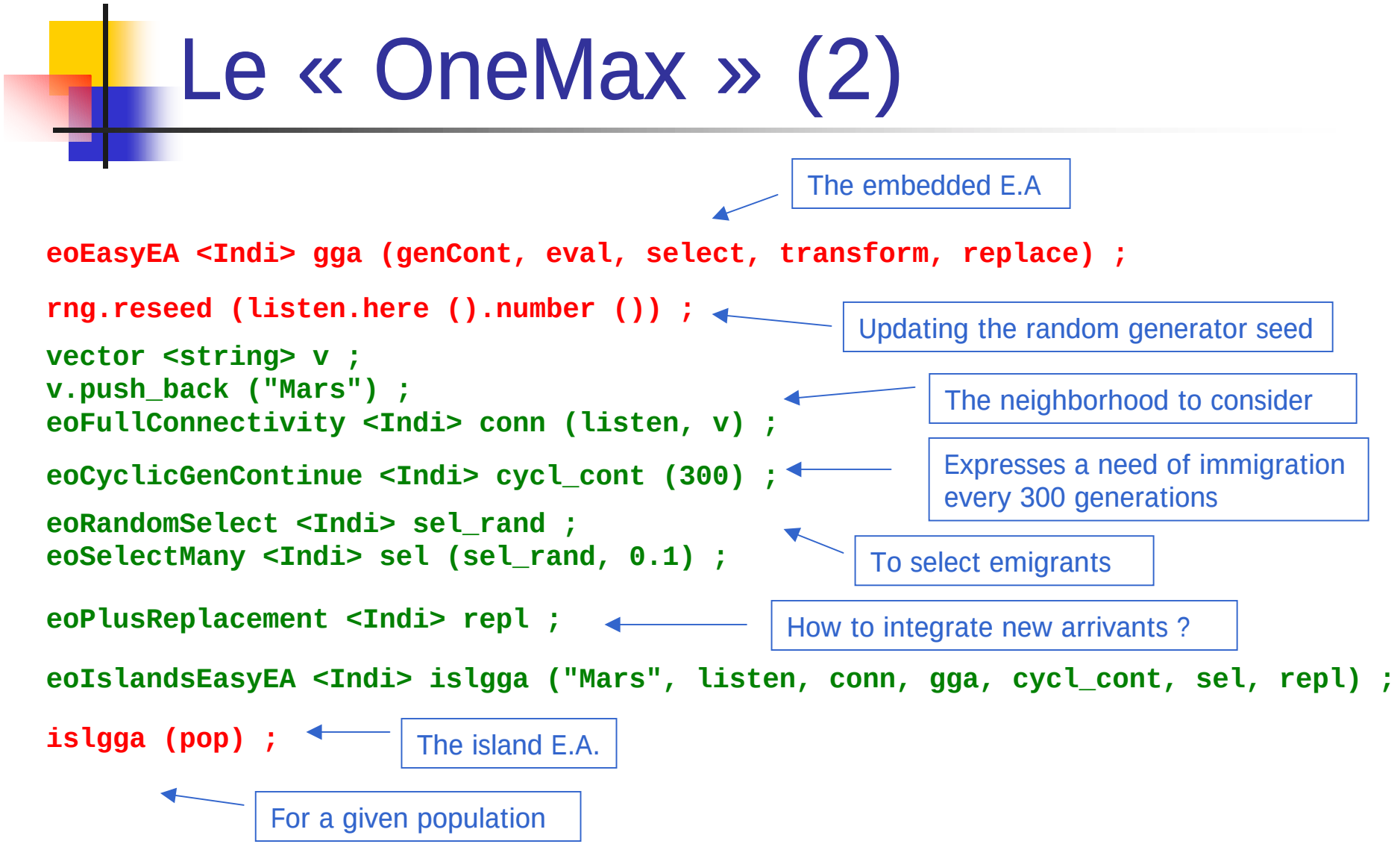
An example : The easy problem of the « OneMax »

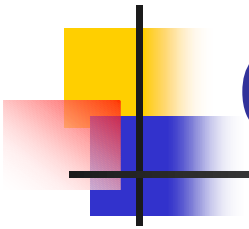
Aim : To maximize the number of 1 in binary strings

The « IslandsEA.cpp » file

```
eoUniformGenerator <bool> uGen ;
eoInitFixedLength <Indi> random (VEC_SIZE, uGen) ;

eoEvalFuncPtr <Indi, double, const vector <bool> & > eval (binary_value) ;
eoPop <Indi> pop (POP_SIZE, random) ;
apply <Indi> (eval, pop) ;

eoDetTournamentSelect <Indi> selectOne(T_SIZE) ;
eoSelectPerc <Indi> select (selectOne) ;

eoGenerationalReplacement<Indi> replace ;

eo1PtBitXover <Indi> xover ;
eoBitMutation <Indi> mutation (P_MUT_PER_BIT) ;
eoSGATransform <Indi> transform (xover, P_CROSS, mutation, P_MUT) ;

eoGenContinue<Indi> genCont (MAX_GEN) ;
```

Le « OneMax » (2)


The embedded E.A
eoEasyEA <Indi> gga (genCont, eval, select, transform, replace) ;
rng.reseed (listen.here ().number ()) ;
Updating the random generator seed
vector <string> v ;
v.push_back ("Mars") ;
eoFullConnectivity <Indi> conn (listen, v) ;
The neighborhood to consider
eoCyclicGenContinue <Indi> cycl_cont (300) ;
Expresses a need of immigration every 300 generations
eoRandomSelect <Indi> sel_rand ;
eoSelectMany <Indi> sel (sel_rand, 0.1) ;
To select emigrants
eoPlusReplacement <Indi> repl ;
How to integrate new arrivants ?
eoIslandsEasyEA <Indi> islgga ("Mars", listen, conn, gga, cycl_cont, sel, repl) ;
islgga (pop) ;
The island E.A.
For a given population



Compilation & run

- Communication layer : MPICH
- For an homogeneous model
- Compilation

  → `mpiCC -I. –I${HOME}/src -c IslandBitEA.cpp`

  → `mpiCC -o IslandBitEA IslandBitEA.o ${HOME}/src/utils/libeoutils.a ${HOME}/src/libeo.a`

- To build a model of 5 islands

  → `mpirun –np 5 IslandBitEA`

Heterogenous models

- Cooperation of heterogeneous E.A. as regards the genetic operators used.
- A new program

The « IslandsEA1.cpp » file

```
eo1PtBitXover <Indi> xover ;
eoBitMutation <Indi> mutation
(P_MUT_PER_BIT) ;

vector <string> v ;
v.push_back ("Mars1") ;
v.push_back ("Mars2") ;
…
eoIslandsEasyEA <Indi> islgga
("Mars1", listen, conn, gga,
cycl_cont, sel, repl) ;
…
```

The « IslandsEA2.cpp » file

```
eoUBitXover<Indi> xover ;
eoDetBitFlip<Indi> mutation ;
…

vector <string> v ;
v.push_back ("Mars1") ;
v.push_back ("Mars2") ;
…
eoIslandsEasyEA <Indi> islgga
("Mars2", listen, conn, gga,
cycl_cont, sel, repl) ;
…
```

An example

- Model : 2 « IslandEA1 » + 3 « IslandEA2 »
- Need of a configuration file looking like

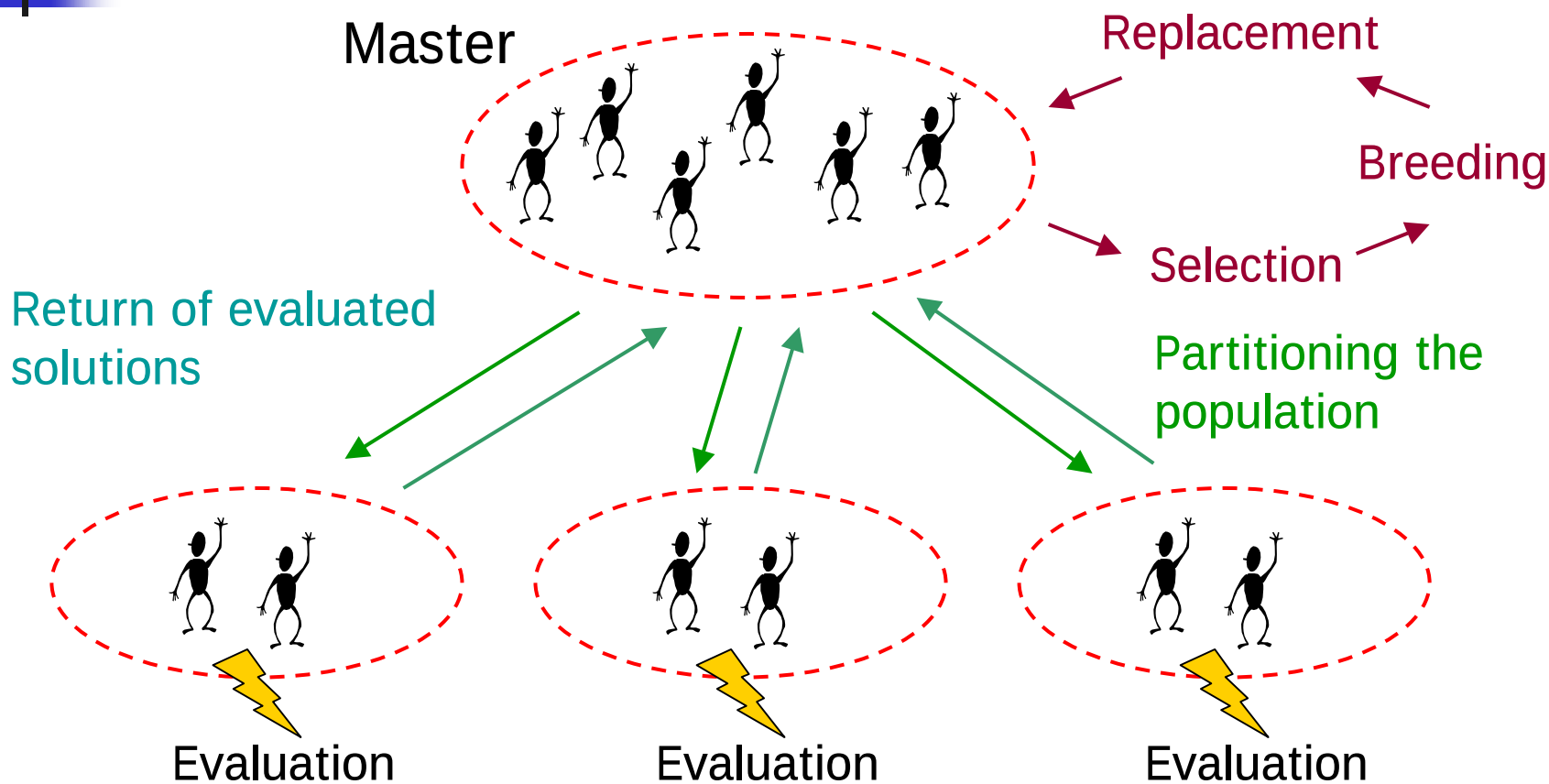
  « IslandEA.conf »

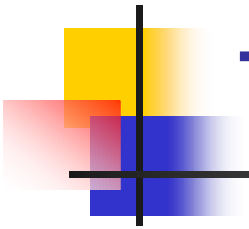
```
127.0.0.1 0 ${EOHOME}/tutorial/island/IslandEA1
wolin.eudil.fr 1 ${EOHOME}/tutorial/island/IslandEA1
gotland.eudil.fr 1 ${EOHOME}/tutorial/island/IslandEA2
124.12.45.97 1 ${EOHOME}/tutorial/island/IslandEA2
124.12.45.98 1 ${EOHOME}/tutorial/island/IslandEA2
```

- Run

```
→ mpirun –p4pg IslandEA.conf IslandEA1
```

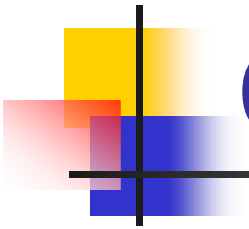
The parallel evaluation : A first example





The parallel evaluation

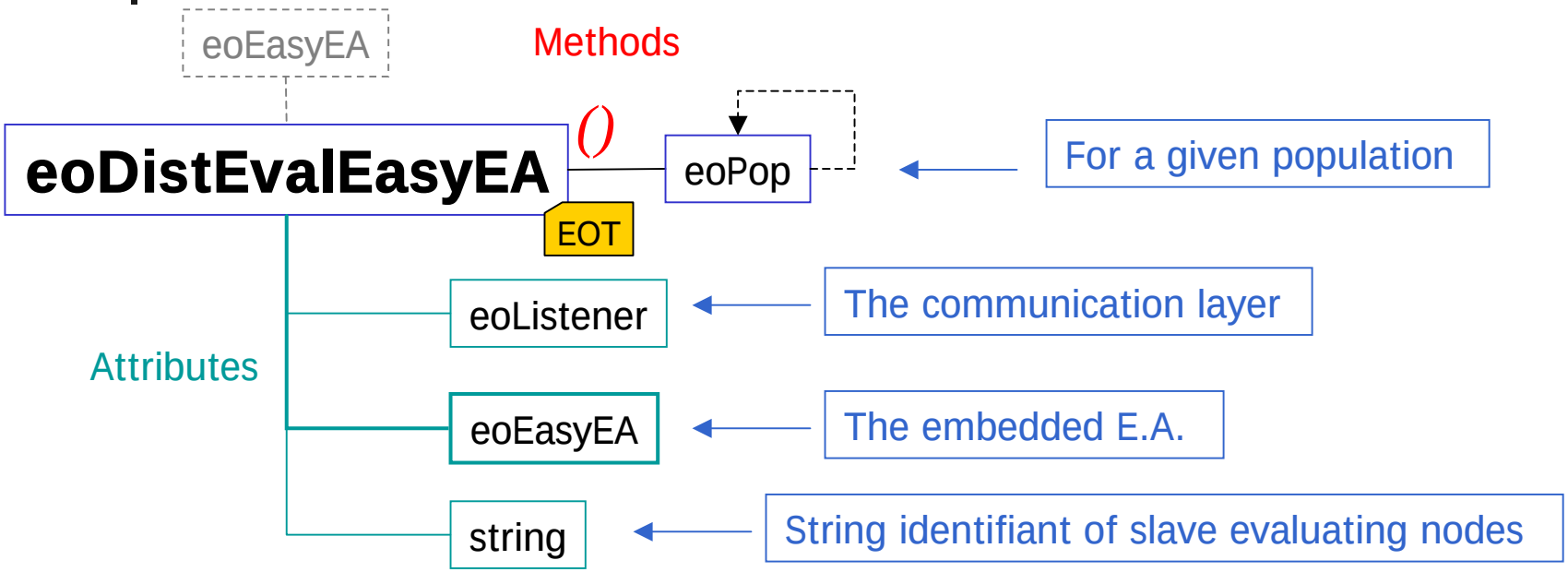
- Why ?
  - Evaluation processes often need high computation time
- A centralized master/slave model
  - Master
    - Performs the selection/transforming/replacement processes,
    - Distributes its population among the known evaluator nodes
  - Slave
    - Receives a sub-population,
    - Evaluates this one,
    - Sends it to the master program.



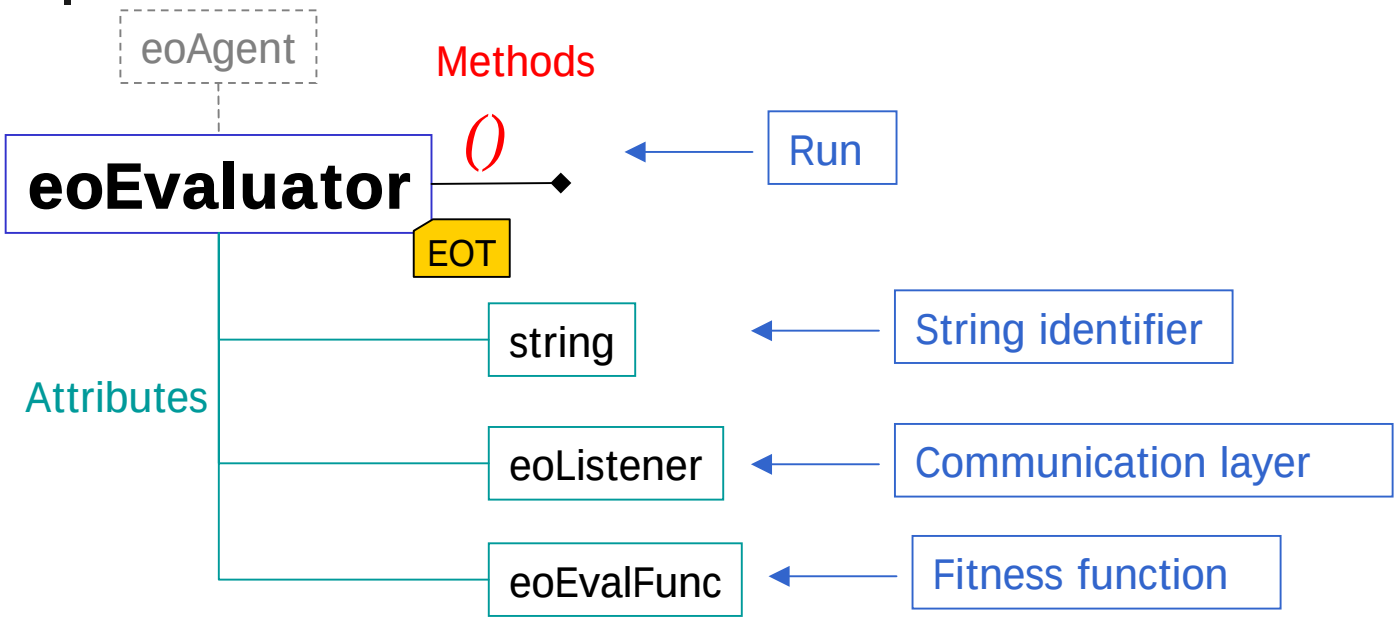
Contribution

- It doesn’t alter the initial E.A.
  - Same results should be obtained more quickly
- A good speed-up if the communication / evaluation ratio is high

The parallel evaluation model in EO



The parallel evaluation model in EO (2)



- Notion of « agent »
  - Enables a service
  - Permanently runs
  - Waiting for the arrival of new populations

An example : The« OneMax » again

The « MasterDistEvalEA.cpp » file

```
…
eoEasyEA<Indi> gga (genCont, eval,
select, transform, replace);

eoListener <Indi> listen
(argc, argv) ;

eoDistEvalEasyEA <Indi> dist_gga
(listen, gga, "Mars") ;

dist_gga (pop) ;

listen.destroy ("Mars") ;
```

To destroy evaluating nodes

The « SlaveDistEvalEA.cpp » file

```
…
eoEvalFuncPtr <Indi, double, const
vector <bool> & > eval
(binary_value) ;

eoListener <Indi> listen
(argc, argv) ;

eoEvaluator <Indi> evaluator
("Mars", listen, eval) ;

evaluator () ;
```

To launch the evaluating node

The run

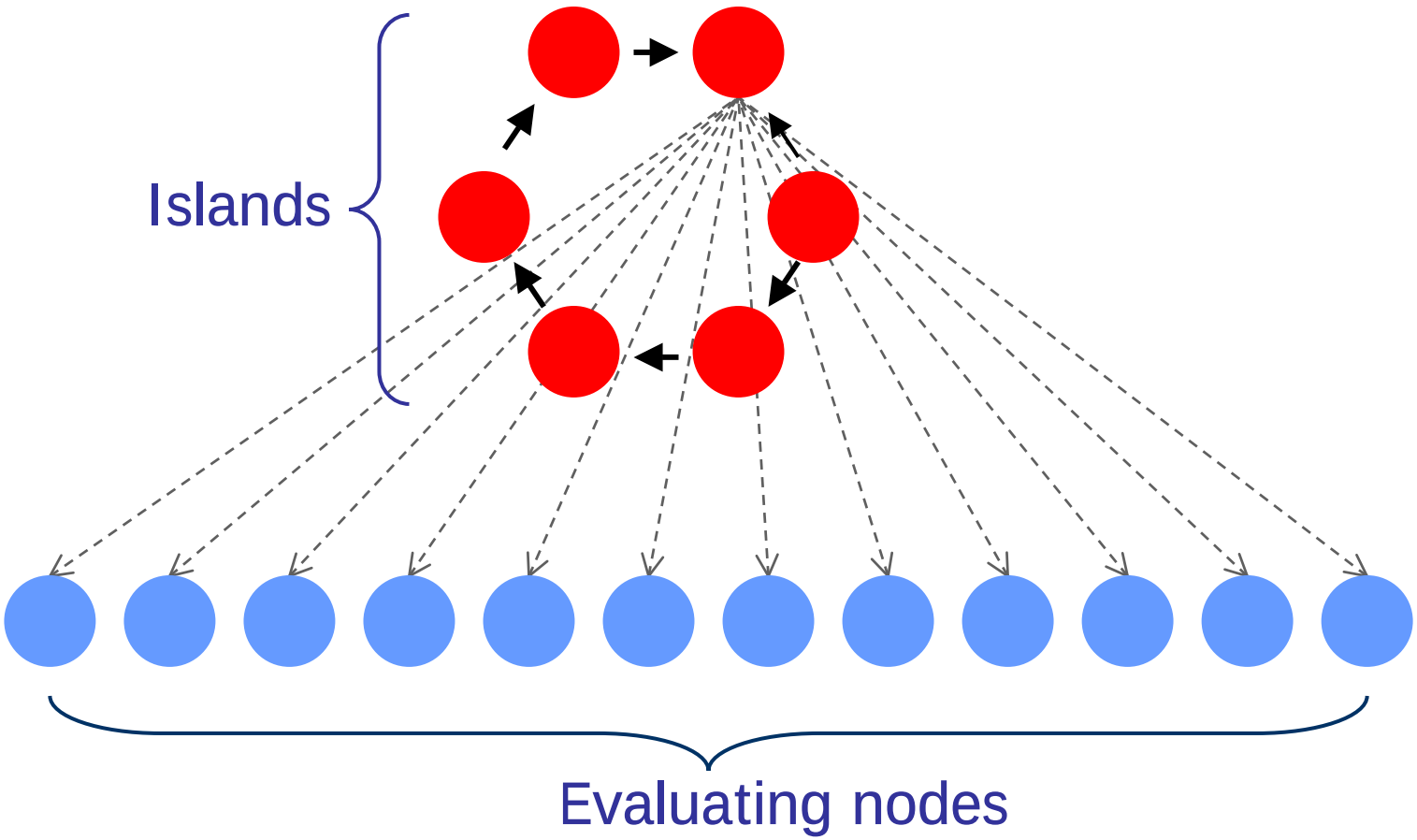
- A model with 4 evaluating nodes
- A configuration file looks like

```
« DistEvalEA.conf »
localhost 0 ${EOHOME}/tutorial/eval/MasterDistEvalEA
localhost 1 ${EOHOME}/tutorial/eval/SlaveDistEvalEA
localhost 1 ${EOHOME}/tutorial/eval/SlaveDistEvalEA
localhost 1 ${EOHOME}/tutorial/eval/SlaveDistEvalEA
localhost 1 ${EOHOME}/tutorial/eval/SlaveDistEvalEA
```

- Run

```
→ mpirun –p4pg DistEvalEA.conf MasterDistEvalEA
```

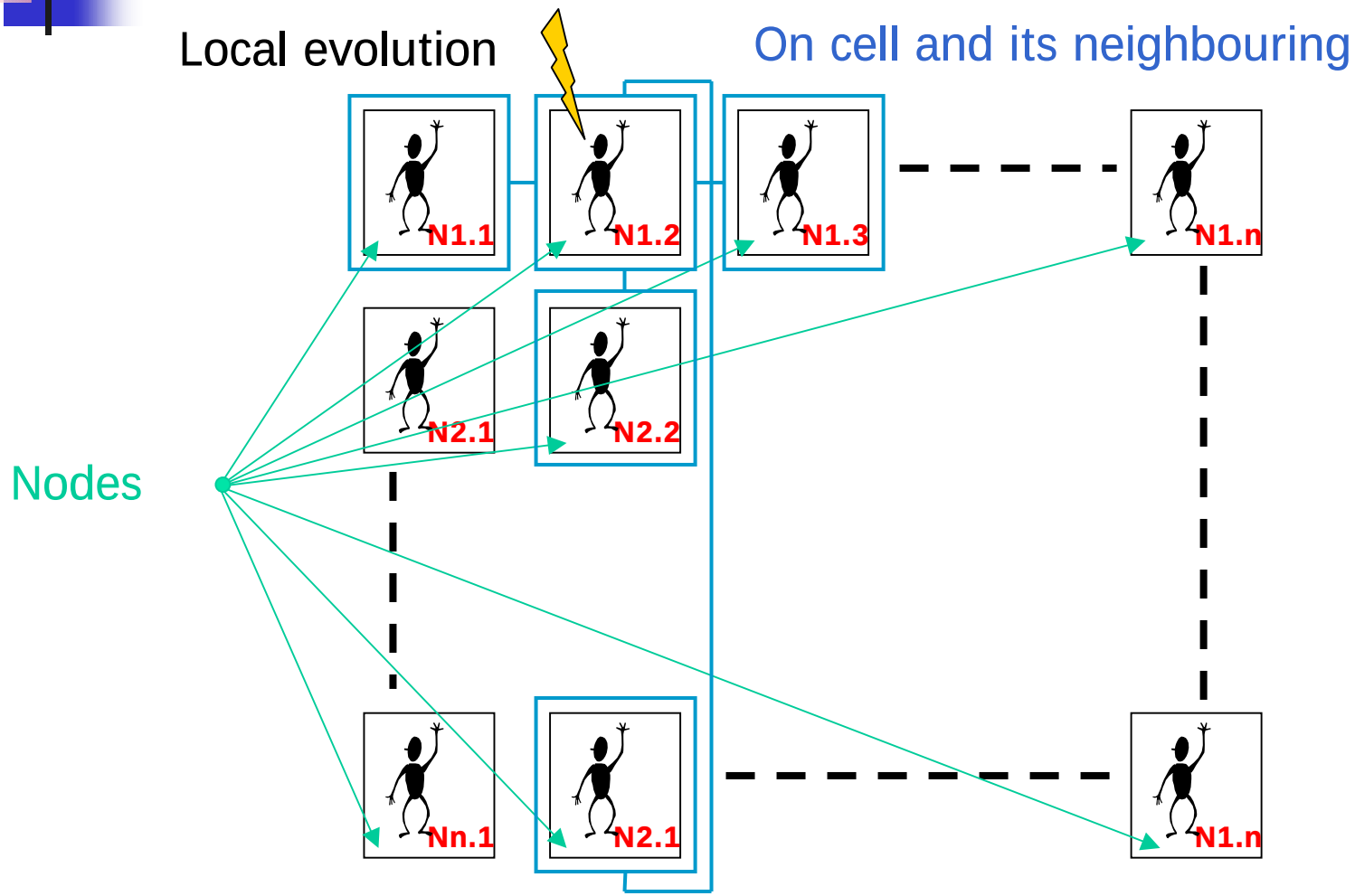
Towards the design of hybrid models



On the previous example

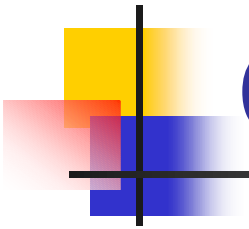
```
eoEasyEA <Indi> gga (genCont, eval, select,
transform, replace) ;
…
eoIslandsEasyEA <Indi> islgga ("Mars",
listen, conn, gga, cycl_cont, sel, repl) ;
…
eoDistEvalEasyEA <Indi> dist_gga (listen,
islgga, "Jupiter") ;

dist_gga (pop) ;
```

The cellular model : How does it work on a toric grid ?



The cellular model

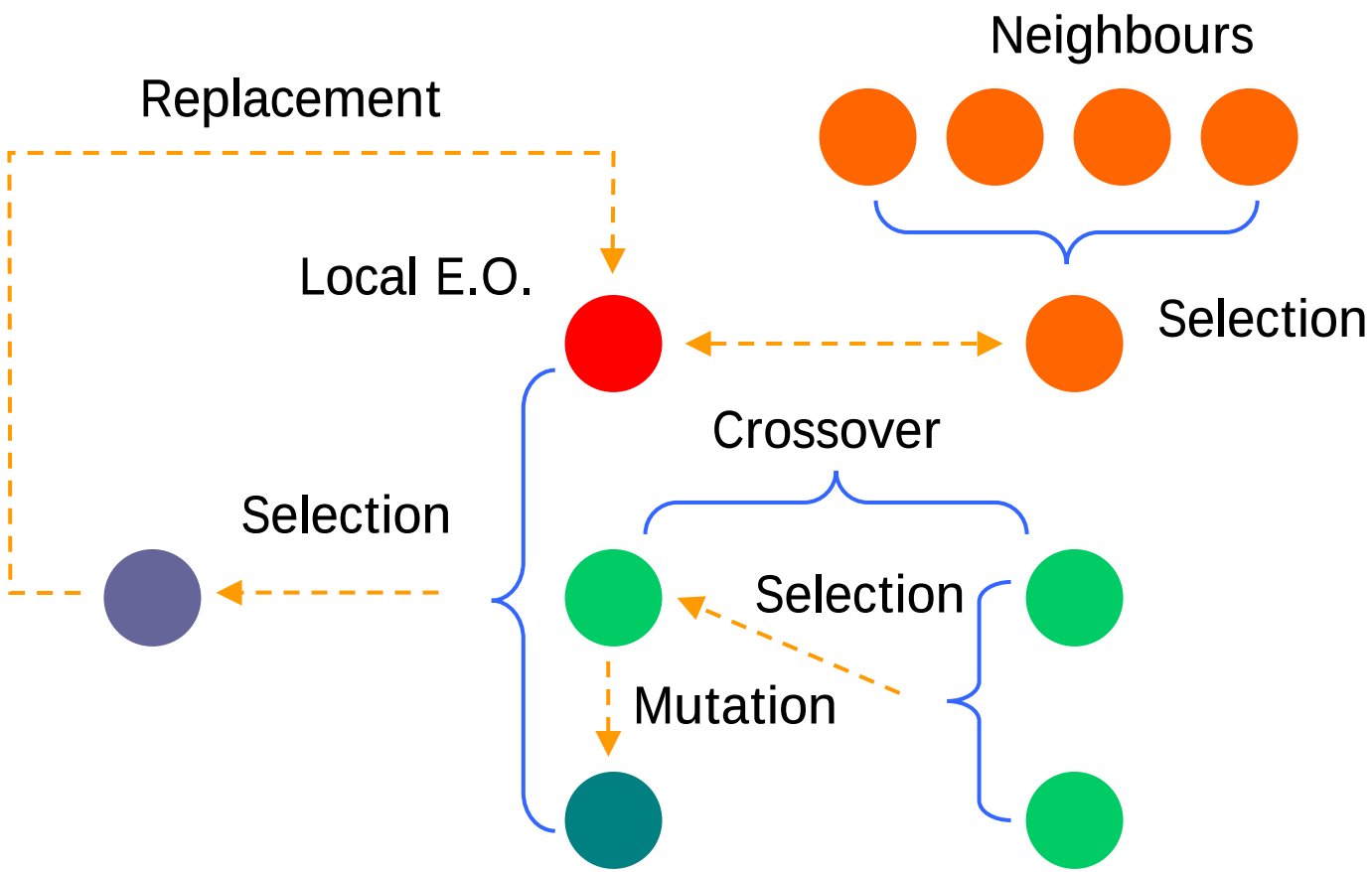
- A totally distributed approach
  - Only one distributed population on a grid of nodes (One solution/ node)
  - Notion of neighborhood
    - Each solution evolves regarding to its neighbouring
- The local work
  - Selection of an individual among the neighborhood
  - Crossover / Mutation
  - Replacement



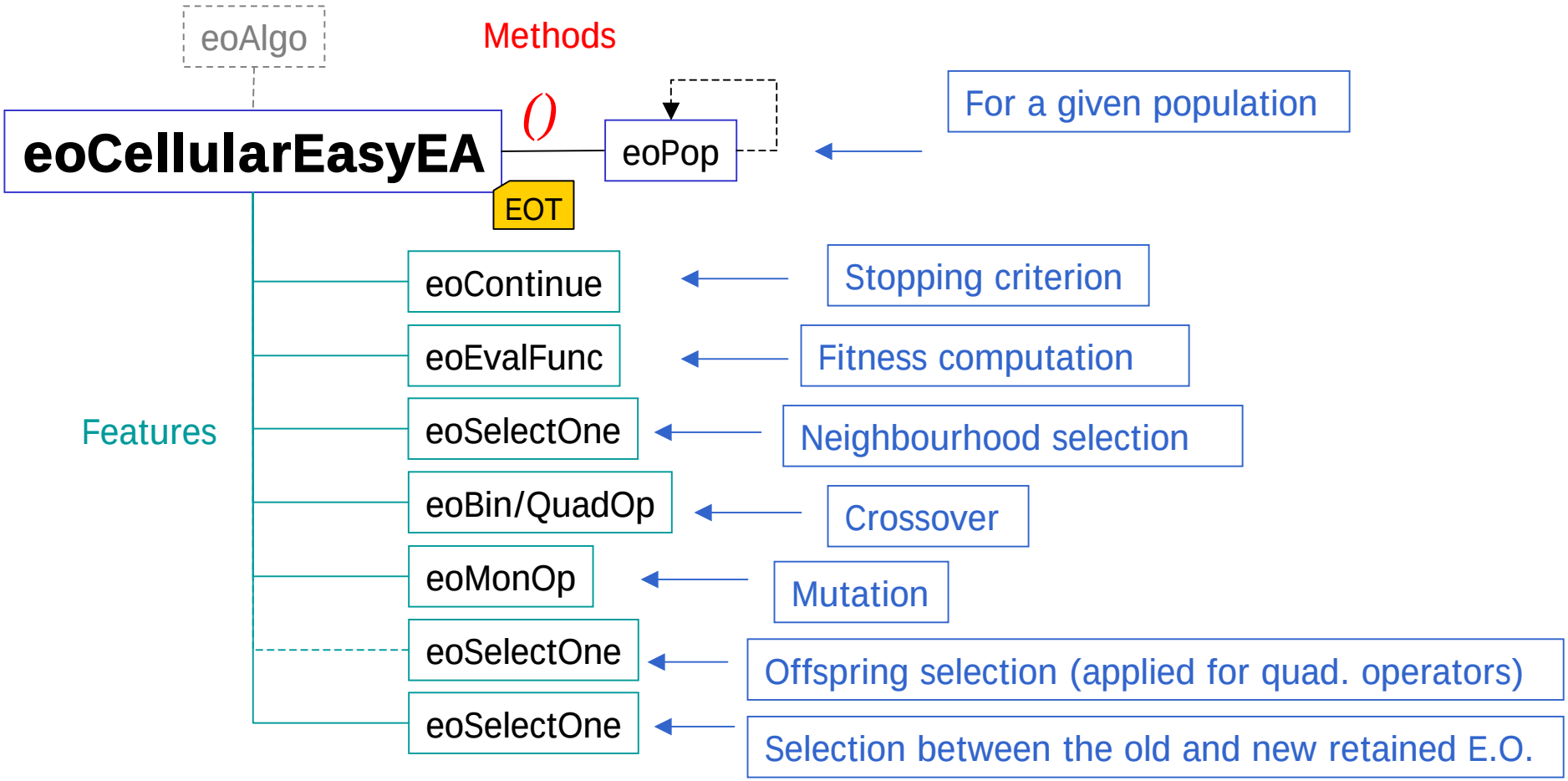
Contribution

- A particularly well-fitted model to massively parallel architectures (fine granularity)
  - Less and less used in favor to SMP et COW/NOW
- Still used on sequential computers
  - Local selection involves a slower convergence, and also a best exploration of the search space

A diagram of a local evolution



The cellular model with EO



The « OneMax » problem again

The « CellularEA.cpp » file

```
eo1PtBitXover<Indi> xover ;

eoBitMutation<Indi> mutationBitFlip (P_MUT_PER_BIT);

eoGenContinue<Indi> genCont(MAX_GEN);

eoBestSelect <Indi> select ;

eoToroidalCellularEasyEA <Indi> gga (genCont,
                                     eval,
                                     select,
                                     xover1,
                                     mutationBitFlip,
                                     select,
                                     select) ;

gga(pop);
```

Quadratic recombination operator

Mutation

The continuation criterion

Elitist selection

Selection applied to the offspring

Selection between the old/new E.O.

Applied for a square-size population