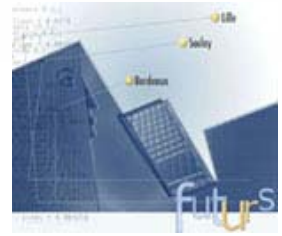
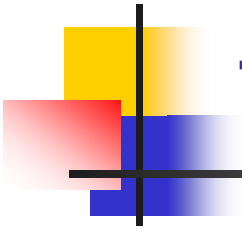
Parallel Cooperative Optimization Research Group

Tutorial EO (Evolving Objects): Using EO for the TSP

Jean-Charles BOISSON

Laboratoire d'Informatique Fondamentale de Lille





Tutorial

- An Evolutionary Computation library
  - ➔ Evolving Objects (EO)
- Tutorial application (Illustration)
  - ➔ The Traveling Salesman Problem (TSP)

Getting started

cd ~/TP_EO_MO

➔ cd TP_EO

➔ ./myautogen.sh

➔ make

➔ cd tsp

➔ gen_algo bench/eil101.tsp

gen_algo.cpp ⇔ main (to modify).

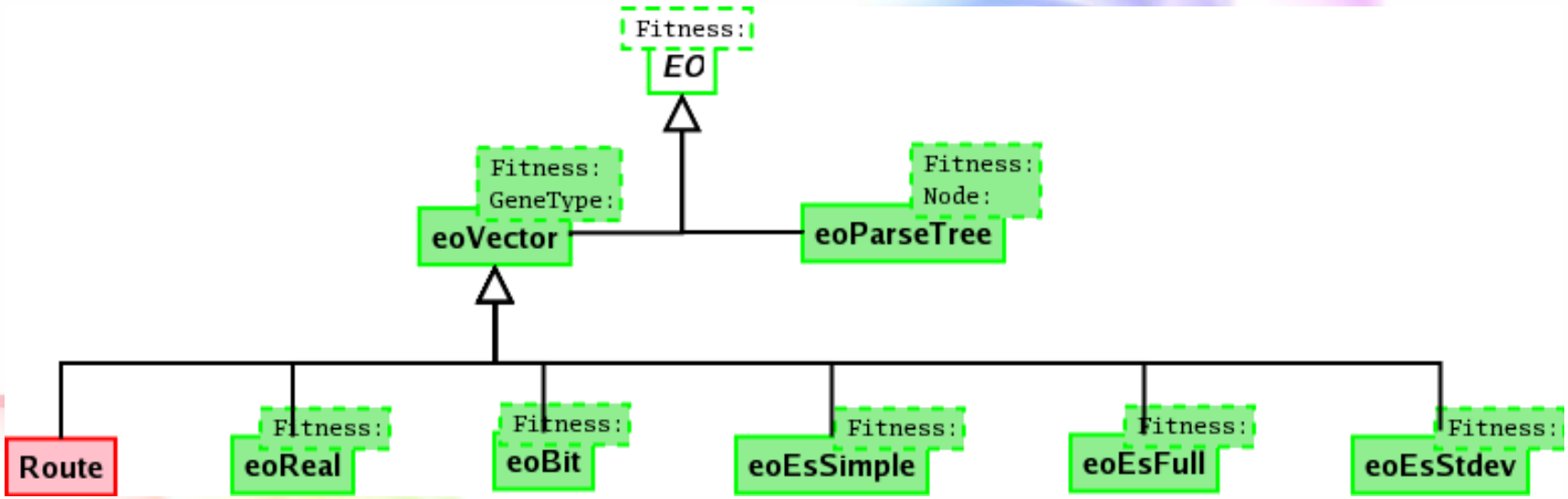
share/ repertory ⇔ tsp « boxes ».

bench/ repertory ⇔ tsp instances.

EO sources:

➔ /home/exterieur/paradise/peo/eo/src

TSP representation



Specific representation

A Genetic algorithm (GA) implementation

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoGenContinue <Route> cont (1000); /* A fixed number of iterations */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);

ea (pop);   /* Application to the given population */
```

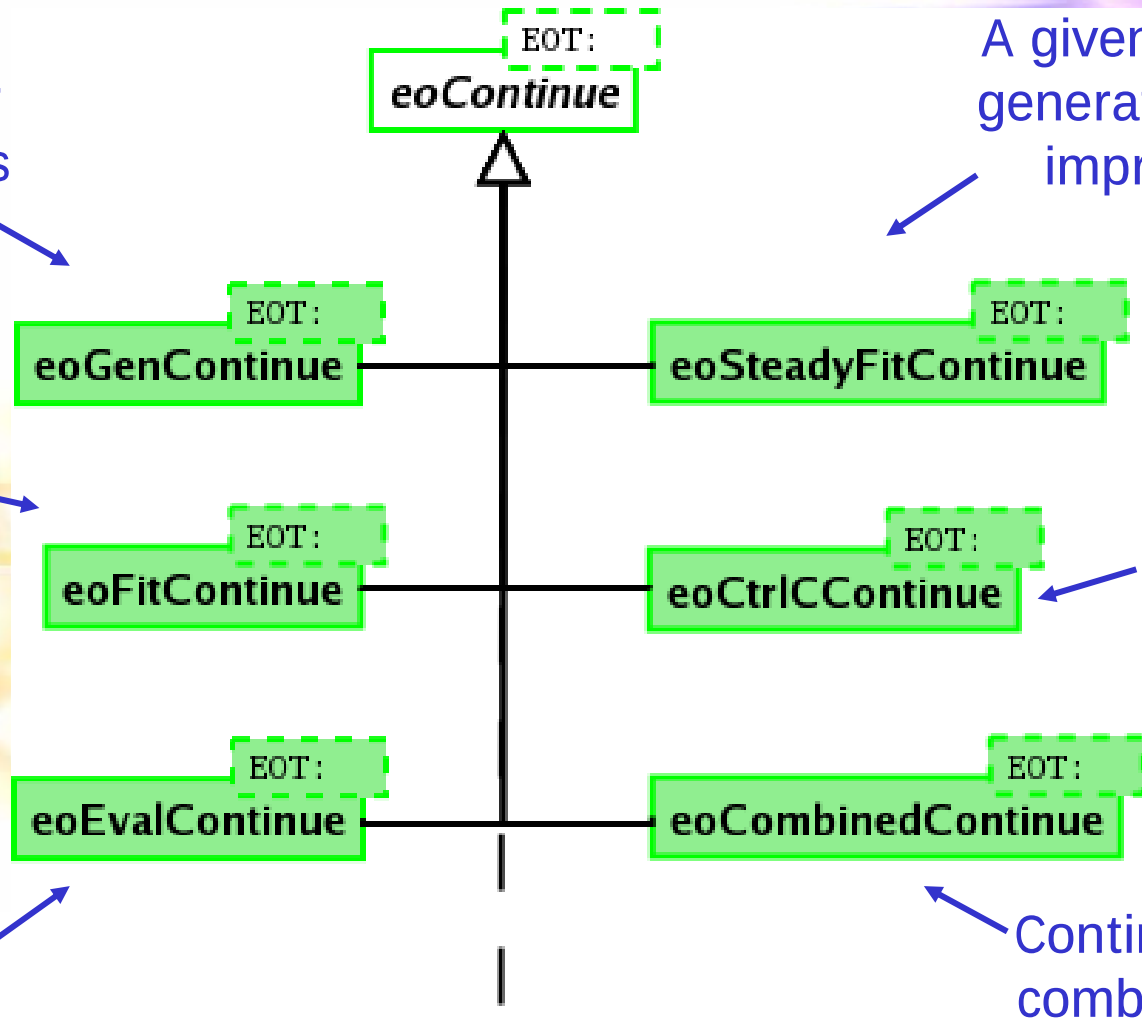
Population size

number of selected individuals

crossover rate

mutation rate

Stopping criteria



Fixed number of generations

A given number of generation without improvement

A threshold fitness is reached

Interactive

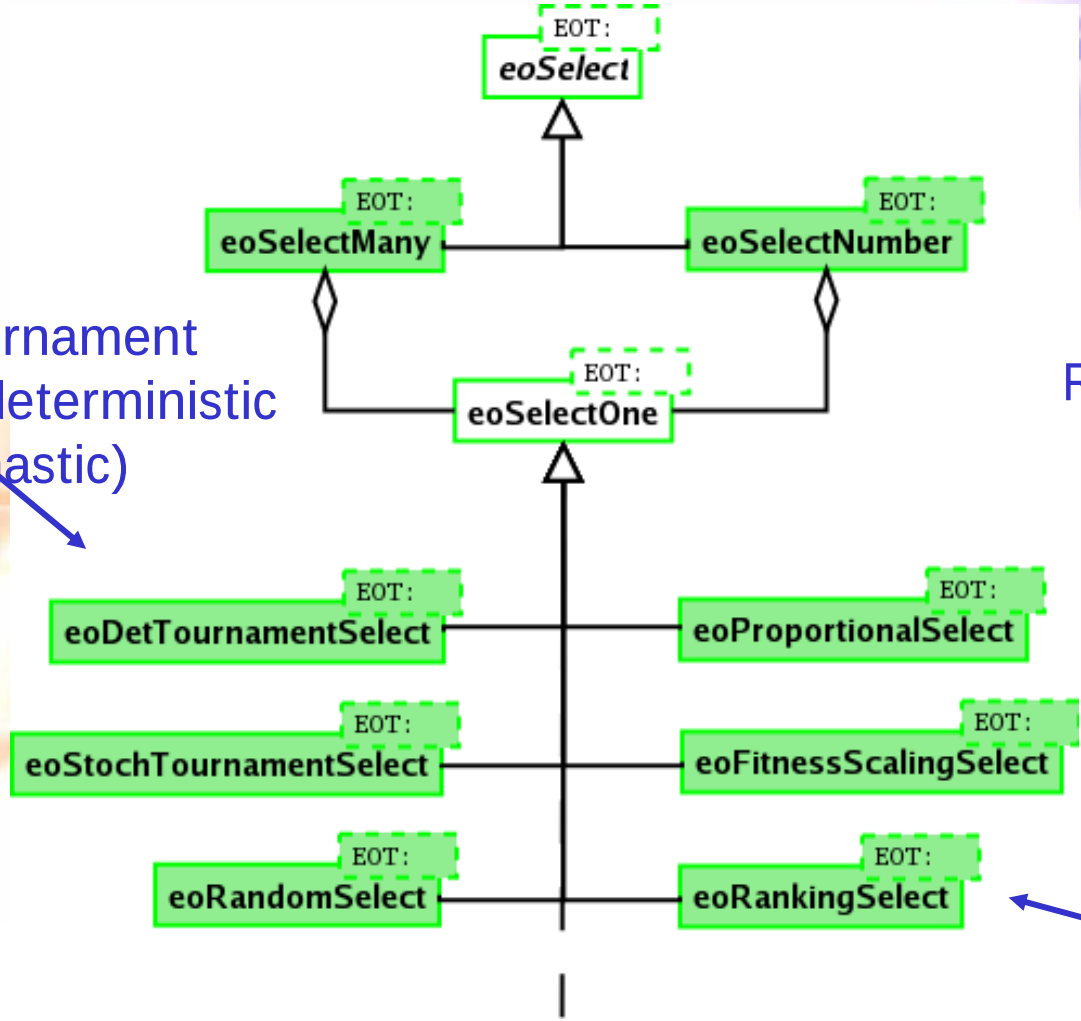
A number of evaluation has been performed

Continuator combination

A GA implementation: a first modification ⇔ stopping criterion

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoSteadyFitContinue <Route> cont (50,100); /* stop when 100 generations have
been made without improvement after having reached a fitness of 50 */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);
ea (pop);   /* Application to the given population */
```

Individual selection strategies



Misc. tournament strategies (deterministic /stochastic)

Roulette wheel

Fitness scaling

Ranking

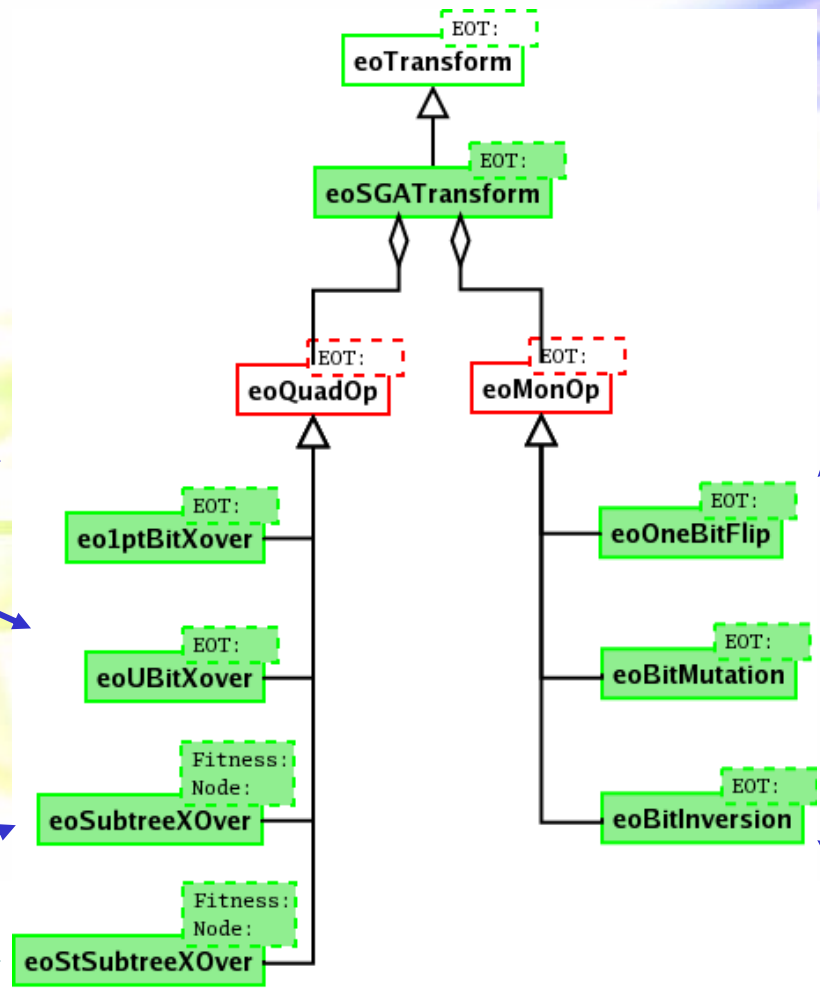
Selection strategies

- NP ⇔ no parameters, indi ⇔ individual.
- eoSelectOne
  - eoBestSelect ⇔ NP, select best indi.
  - eoDetTournamentSelect ⇔ tournament size (>=2).
  - eoEliteSequentialSelect ⇔ already sorted or not, best indi selected.
  - eoNoSelect ⇔ NP, indi fitness not used, indis return in order.
  - eoProportionalSelect ⇔ NP, roulette wheel selection.
  - eoRandomSelect ⇔ NP, random selection.
  - eoSequentialSelect ⇔ already sorted or not, indi selected in order.
  - eoStochasticUniversalSelect ⇔ NP, ordered according to one random number.
  - eoStochTournamentSelect ⇔ tournament rate (> 0.5 and <= 1).
  - ...

A GA implementation: 2nd modification ⇔ selection strategy

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoGenContinue <Route> cont (1000); /* A fixed number of iterations */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);

ea (pop);   /* Application to the given population */
```

Transformations: generic operators



1-point crossover

Uniform crossover

Tree-based crossovers

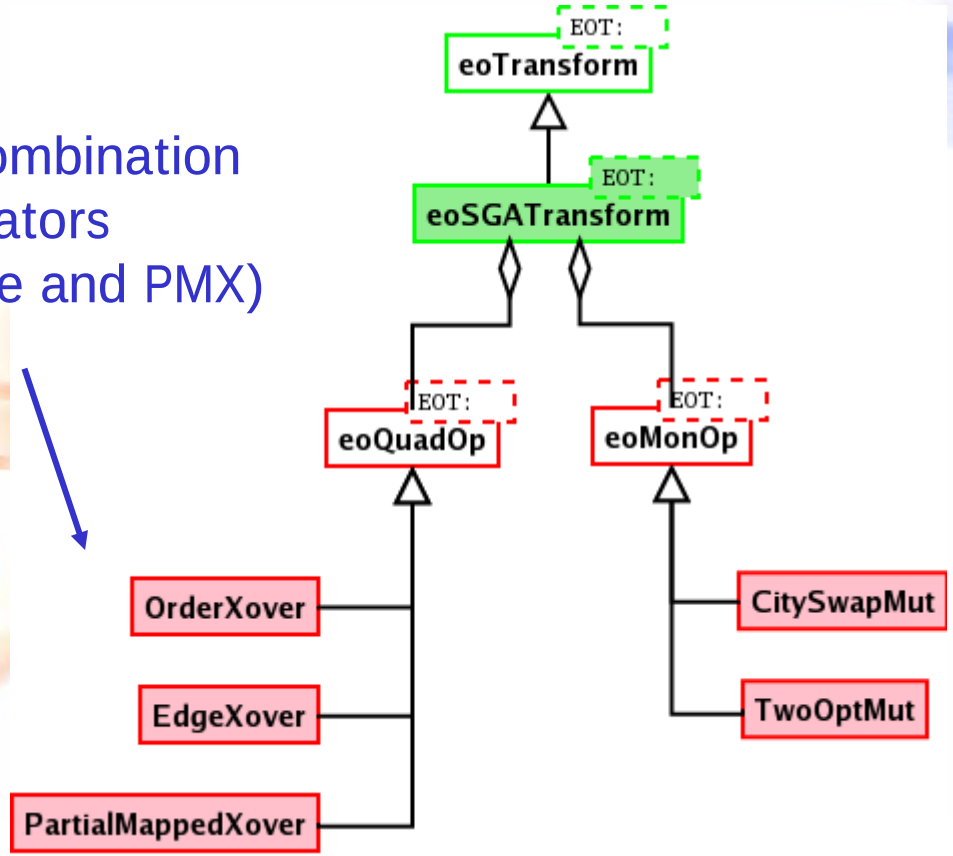
A random bit is mutated

Some bits are mutated

Inverse the bits between an interval

Transformations: TSP crossovers and mutations

Three recombination operators (order, edge and PMX)



Two mutation operators (City-swap and two-opt)

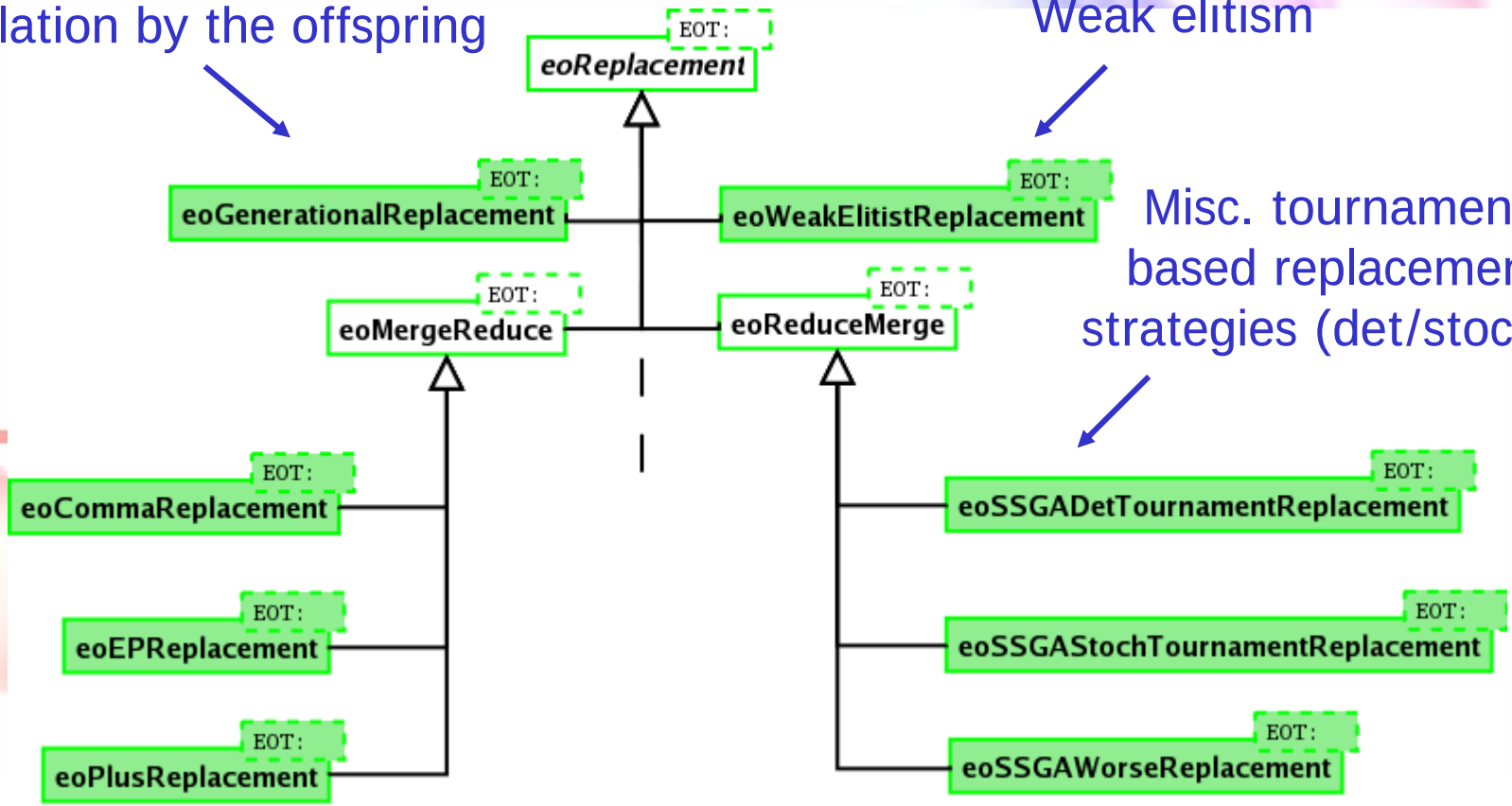
A GA implementation: 3rd modification ⇔ operators

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoGenContinue <Route> cont (1000); /* A fixed number of iterations */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);

ea (pop);   /* Application to the given population */
```

Replacement strategies

Replacement of the whole population by the offspring

Weak elitism



Misc. tournament based replacement strategies (det/stoch.)

Strong elitism

Create a replacement

⇔ merge + reduce

- eoPlusReplacement:
  - eoPlus.
  - eoTruncate.
- eoCommaReplacement:
  - eoNoElitism.
  - eoTruncate.
- eoEPReplacement:
  - eoPlus.
  - eoEPReduce.
- ...

Merging Strategies

- eoMerge
  - eoElitism ⇔ the number or the rate of the best individuals to add in the offspring.

    ➔ offspring = offspring + N best parents

    with N = number given or rate given * parents size.
  - eoNoElitism ⇔ NP, offspring = only offspring.
  - eoPlus ⇔ NP, offspring = offspring + parents.

A GA implementation: 4th modification ⇔ merging strategy

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoGenContinue <Route> cont (1000); /* A fixed number of iterations */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);

ea (pop);   /* Application to the given population */
```

Reduction Strategies

- eoReduce
  - eoDetTournamentTruncate ⇔ rate (> 0.5 and <= 1), truncate individuals by inverse deterministic tournament.
  - eoEPReduce ⇔ size (>= 2), truncate individuals according to an EP tournament.
  - eoLinearTruncate ⇔ NP, truncate worst individuals.
  - eoRandomReduce ⇔ NP, the individual are shuffle and the population is truncated.
  - eoStochTournamentTruncate ⇔ rate (> 0.5 and < 1), truncated individuals according to a inverse stochastic tournament.
  - eoTruncate ⇔ truncate a population to a size < current size.

A GA implementation: 5th modification ⇔ reduction strategy

```
RouteInit route_init; /* It builds random routes */
RouteEval full_route_eval; /* Full route evaluator */

eoPop <Route> pop (100, route_init);  /* Population */
eoGenContinue <Route> cont (1000); /* A fixed number of iterations */

 eoStochTournamentSelect <Route> select_one; /* Stoch. Tournament selection */
eoSelectNumber <Route> select (select_one, 100);

OrderXover cross; /* Recombination */
CitySwap mut;  /* Mutation */

/* Standard SGA Transformation */
eoSGATransform <Route> transform (cross, 1, mut, 0.01);

eoElitism <Route> merge(1); /* merging parents offspring */
eoStochTournamentTruncate <Route> reduce (0.7); /* Tournament reduce. */

eoEasyEA <Route> ea (cont, full_eval,select, transform,merge,reduce);

ea (pop);   /* Application to the given population */
```

Other features (1/2)

- Boxes combination:
  - Stopping criterions:
    - eoCombinedContinue ⇔ logical AND between the criterions.
  - Checkpointing system:
    - Additional processes between each generation.
    - Print and save statistics (nb gen, best fitness, ...).
    - Visualization
- Boxes extension:
  - New template objects.
  - New dedicated objects.

Other features (2/2)

- Parameters management:
  - Configuration files.
  - Editing and parsing systems

    ➔ included in EO.
- Fault tolerant system:
  - A population can be saved (according to user specifications).

    ➔ a run can be stopped and continued later.

Conclusions

- Thanks to EO, you have:
  - Designed your own genetic algorithm for the TSP problem.
  - Use and (maybe) create template boxes
    ➔ not linked to one problem.
  - Made only a first step to:
    - Solution exploitation ⇔ MO.
    - Multi-objective problems ⇔ MOEO.
    - Parallelization and hybridization ⇔ PARADISEO.