



Evolving Objects: Yet Another Evolutionary Computation Library?

M. Keijzer¹, J. J. Merelo², G. Romero², M. Schoenauer³

<http://eodev.sourceforge.net/>

Presented at **EA'01** – October 29. 2001



What is EO?

An Open Source STL-based C++ **EC library**

Don't worry, this is a **functional** talk :-)

- Paradigm-free supersedes EP, ES, GP, GA and many more!
- General operators $n \rightarrow m$
- Generic operators Representation independent
- User parameter definition Command-line / parameter file
- Visualization, saving/restarting and **many more!**
- Getting started “facilities” and GUI ...EASEA



The technical slide



Re-entering code

- No global variable :-)
- → co-evolution, meta-evolution, subroutine use, ...

Objects, not functions

STL functions

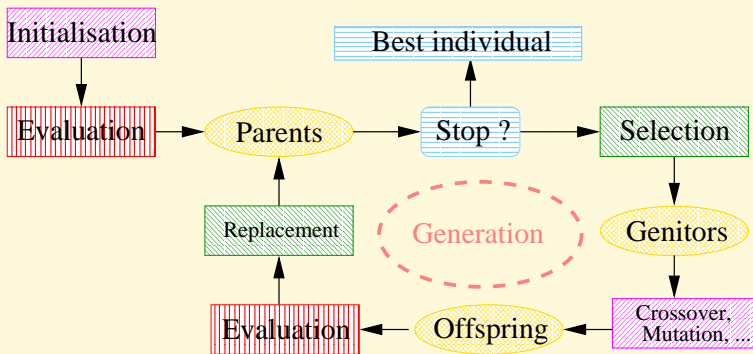
- Class with single interface/method: **operator()**
- Private data, can be passed as a whole





Combination of objects (functors)

- Choosing among different possibilities **at run times**
- A **combined blip is a blip**



An Evolutionary Algorithm

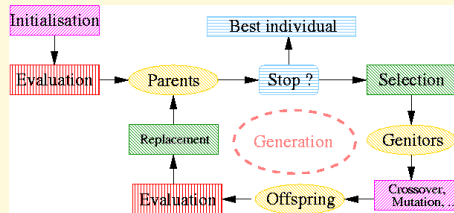


-  Stochastic operators: Representation dependent
-  "Darwinism" (stochastic or determinist)
-  Main CPU cost
-  Checkpointing: stopping criterion and statistics

From EO tutorial



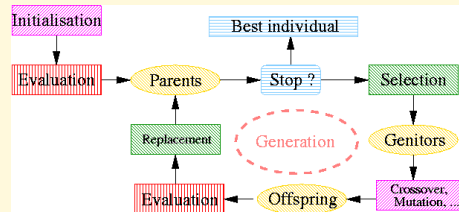
Representation



- **Any** structure Later templated over the fitness
- Use of STL containers vectors, lists, ...
Relieves from (most) memory management
- Easy construct complex structures
and of instantiated **generic** variation operators



Fitness computation

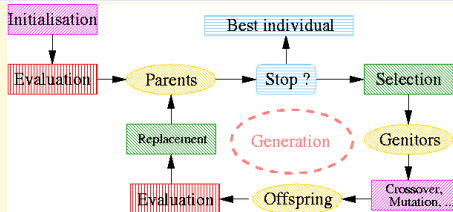


Any fitness type Scalar, scalar with constraints, multi-objective, ...

- The only code to write for existing representations
- No useless computation Through EOs **invalid** flag
- Trivial embedding of plain C-code with correct interface
- Call of existing objects (C, Fortran) with correct interface



Evolution Engine



Selection

- Popular scalar routines

Roulette, ranking, tournament, ...

- Multi-objective

NSGA2

Replacement: A general scheme

1. Selector among parents and children

or none

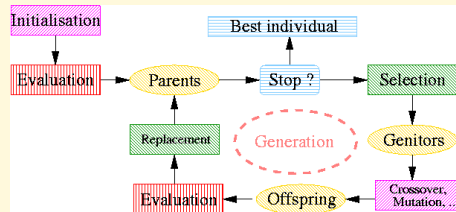
2. \pm Merging with parents

3. \pm Selector among merged population

\pm Weak and strong elitism



Variation operators



- **Objects**, not methods

Encapsulation

- Standard types

- Mutation

Unary, $1 \rightarrow 1$

- Crossovers

Binary, $2 \rightarrow 1$

“Quadratic”, $2 \rightarrow 2$

- General type

$n \rightarrow m$



General operators

$$n \rightarrow m$$

- $n \rightarrow 1$: **Orgy** operators
- $P \rightarrow 1$: **Global** recombination
Uses the whole population as parents
- $0 \rightarrow m$: **Creation** of new individuals
Partial restart, SDM, ...

Embedded in a **breeder** object

that takes care of the number of offspring



Combining operators

Real-world EAs use more than one operator of each kind!

EO solution: combine them

Recursive structure

Proportional combination

- User-defined relative weights
- Roulette-wheel choice
- One and only one will be applied



Combining operators (2)

Sequential combination

- Each operator is applied with its own probability
- From none to all of them can be applied (sequentially)

Supersedes standards

- GA-like using p_{cross} and p_{mut}
- ES-like global recombination + mutation



Generic operators

Bottom-up construction of variation operators
for genotypes based on STL data structures

vector, list, ...

Key issues

- Fixed length vs variable length
- Order-dependent vs order independent

Base ideas for EASEA Graphical genotype builder **forthcoming!**



Generic operators (2)



Crossovers on vectors/lists

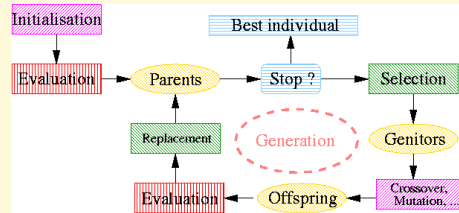
- N-point, uniform Order- and length- preserving
- Call element crossover (fixed rate, or k times) idem
- Exchange with external criterion Order- and length- disturbing

Mutations on vectors/lists

- Call element mutation (fixed rate, or k times) Order- and length- preserving
e.g. binary mutations (bit-flip and “deterministic”)
- Add/remove element Length- disturbing



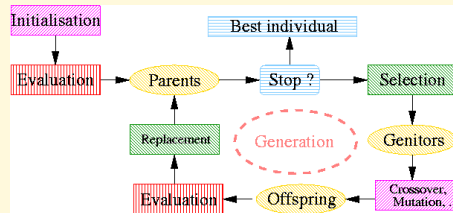
Initialization



- **Initializer objects** Can be combined ... the proportional way
- **Generic** for complex genotypes from base generators
- **Reload** saved state of some previous run
- **Included RNG** Mersenne Twister
 - Results independent of hardware/OS Reproducibility
 - Run replay For debugging!
 - “True” randomness Using hardware clock



Checkpointing



What you want to do **every generation**

- Check stopping criteria **is a combined stop criterion**
- Compute and monitor statistics
Average and best, diversity, rate of feasible guys, ...
- Save current state **see Initialization**
- Update whatever needs to be updated
e.g. dynamic or adaptive population level parameters



Statistics

Computing statistics

- Write the code into a **stat** class
- Create a **stat** object , “add” it to the **checkpoint**
- It will receive the current population every generation

Monitoring statistics

- Create a monitor (once!) Screen (text, gnuplot), file, ...
- Add the monitor to the **checkpoint** (once!)
- Add **stat** objects to the monitor
- Will be displayed/saved to disk every generation



Updates

Dynamic parameter: example

Class **Blip** has a static parameter.

- Derive a class **NewBlip** from both **Blip** and **Update**
- Write the parameter update in the **update** method
- Use **NewBlip** in place of **Blip**
- “Add” that **NewBlip** object to the **checkpoint**

The **update** method will be called every generation



Cons

- Huge compilation times Linux and g++
- CygWinTM required on MicrosoftTM systems
- Getting started ... **but**
 - A tutorial (on-going :-(
Download ready-to-use snapshot at Polytechnique
 - Shell scripts for new representations
 - **EASEA** :-)



On-going

- Multi-objective as user-friendly as scalar
CMAP et Fractales
- Constraint handling methods CMAP et Fractales
- Parallelization Master-slave, island model, distributed population
Granada et LIFL
- Port to Java within European DREAM project



Conclusion

Available representations

GAs, self-adaptive ESs, GP (symbolic regression), Voronoi.

Pros summary

- Highly flexible
- Not-so-difficult to use :-)
- Complete GUI forthcoming

It's free software: Use it – and **contribute!**