Description of Evolution Engine Parameters

Luis Da Costa and Marc Schoenauer

April 15, 2008

Chapter 1

Description of Evolution Engine Parameters

Description The Evolution Engine (EE) of an EA contains the parameters of the engine shared by any kind of algorithm. They are described by an XML file (see Table 1.1)

```
<EVOLUTION_ENGINE>
  <P_MUT>0.1</P_MUT>
  <P_CROSS>0.85</P_CROSS>
  <MIN_MAX>Minimise</MIN_MAX>
  <NUM_GEN activated="true">500</NUM_GEN>
  <NUM_EVAL activated="false">1</NUM_EVAL>
  <MAX_TIME activated="false">1</MAX_TIME>
  <GRAPH_STAT>true</GRAPH_STAT>
  <ENGINE>SteadyState GA</ENGINE>
  <POP_SIZE>100</POP_SIZE>
  <ELITE TYPE="perc">15</ELITE>
  <FERTIL TYPE="perc">85</FERTIL>
  <NB_GEN TYPE="perc">100</NB_GEN>
  <GEN_SEL PARAM="2.0">Tournam.</GEN_SEL>
  <NB_OFFSPR TYPE="perc">100</NB_OFFSPR>
  <SURVIVING_OFFSPR TYPE="perc">100</SURVIVING_OFFSPR>
  <SURVIVING_PARENTS TYPE="perc">25</SURVIVING_PARENTS>
  <RED_PARENT PARAM="2.0">E.P. Trn.</RED_PARENT>
  <RED_OFFSPR PARAM="2.0">E.P. Trn.</RED_OFFSPR>
  <FINAL_RED PARAM="2.0">Sequent.</FINAL_RED>
  <ELITISM>Strong</ELITISM>
</EVOLUTION_ENGINE>
```

Table 1.1: XML Structure of an Evolution Engine (EE)

Each one of the tags of the XML file represents one parameter of the engine. These are all encapsulated in class `guide.kernel.EvolutionEngine`, who then provides methods to access and set the information. An explanation of the parameters is given here:

1. **General Parameters**:

   - Engine specification. GUIDE allows the choosing of a specific flavour of engine. The tag associated is **ENGINE**, and it can take one of these values:

     (a) **Generational GA**: this is probably the most popular GA algorithm. In this mode, all population is fertile, and is then potentially chosen for offspring generation. The new population (offspring) replaces the old population (parents).

     (b) **SteadyState GA**: on each generation, one offspring is generated and replaces one of the parents.

     (c) **Evolution Strategies, ES**: in this engine there is no crossover operator nor selection of the parents; instead, μ parents give birth to λ offspring. There is two flavours for this engine: either the (μ) individuals of the new population are only chosen from the (λ) offspring (in which case we talk about (μ, λ)-ES) or the (μ) individuals of the new population are only chosen from the sum of the old population with the offspring (so $\mu+\lambda$ individuals). In this case we talk about $(\mu + \lambda)$-ES.

     (d) **Evolutionary Programming**: this is basically $(\mu + \lambda)$-ES.

     (e) **Custom**: No restriction on parameters.

   - Problem is minimization or maximization: tag **MIN_MAX**. Value **Minimise** means that the goal is to minimize the fitness function. Value **Maximize** means that the goal is to maximize it.

   - Stopping criteria:

     – Num generations: tag NUM_GEN. Algorithm runs until the specified number of generations is reached. Only taken account if parameter *activated* of the tag is set to *"true"*

     – Num evaluations: tag NUM_EVAL. Algorithm runs until the specified number of evaluations is reached. Only taken account if parameter *activated* of the tag is set to *"true"*

  - Maximum time to run: tag MAX_TIME. Algorithm runs until the time specifed is reached. Only taken account if parameter *activated* of the tag is set to *"true"*

- Feedback to the user: tag GRAPH_STAT. Gives graphical feedback to the user if the value is *true*

2. **Initial Population Parameters**:

   - Population size: tag POP_SIZE. Size of the initial population on each generation.
   - Elite value: tag ELITE. Specifies the number (tag attribute TYPE=*"abs"*) or percentage (tag attribute TYPE=*"perc"*) of the initial population that will intervene directly on the building of the final population.
   - Elite value: tag FERTIL. Specifies the number (tag attribute TYPE=*"abs"*) or percentage (tag attribute TYPE=*"perc"*) of the initial population that is eligible for giving birth to offspring.

3. **Intermediate Population Parameters**:

   - Choosing parents:
     - How many genitors to choose from the fertile: tag NB_GEN. Specified as a number (tag attribute TYPE=*"abs"*) or as apercentage (tag attribute TYPE=*"perc"*).
     - How to choose the genitors from the fertile: tag GEN_SEL. See below for details.
   - Creating offspring:
     - Mutation rate: tag P_MUT.
     - Crossover probability: tag P_CROSS.
     - How many offspring to choose from the genitors: tag NB_-OFFSPR. Specified as a number (tag attribute TYPE=*"abs"*) or as apercentage (tag attribute TYPE=*"perc"*).
   - Building intermediate population:
     - How many individuals to choose from the offspring to build intermediate population. Specified as a number (tag attribute TYPE=*"abs"*) or as apercentage (tag attribute TYPE=*"perc"*).

  - How to choose the individuals from the offspring: tag RED_-OFFSPR. See below for details.
  - How many individuals to choose from the fertile to build intermediate population: tag SURVIVING_PARENTS. Specified as a number (tag attribute TYPE=*"abs"*) or as apercentage (tag attribute TYPE=*"perc"*).
  - How to choose the individuals from the fertile: tag RED_-PARENT. See below for details.

4. **Final Population Parameters**:

   - Is the elite population going directly to the new population (in this case tag ELITISM = "Strong") or does it have to *fight* with the intermediate population (tag ELITISM = "Weak")?
   - How to choose the final population from the intermediate population: tag FINAL_RED. See below for details.

Selectors implemented in GUIDE A description of the Selectors implemented in GUIDE is given here. For a very interesting and detailed study of selection procedures, please see the work of Bäck [**?**]. For each Selection method we give the constant that represents it and its string descriptor (used in between the XML tags on the file description). All constants are defined in `guide.kernel.Util.EvolutionSelectionConstants`

1. *Roulette Wheel.* Represented in GUIDE by the constant `EvolutionSelectionConstants.ROULETTE_WHEEL`. String descriptor: "*Roul. Wh.*"
   Also called *Proportional Selection*, this method was introduced by Holland for the Genetic Algorithm [**?**]. It assigns selection probabilites according to the relative fitness of individuals:

$$p_i = \frac{\Phi(a_i)}{\sum_{j=1}^{\lambda} \Phi(a_j)} \tag{1.1}$$

2. *(Linear) Ranking.* Represented in GUIDE by the constant `RANKING`. String descriptor: "*Ranking*"
   Defined by Baker [**?**], it uses a linear function to map indices i to selection probabilities p_i. Assuming that individuals are sorted according

to their fitness, the selection probabilities are given by

$$p_i = \frac{\eta^+ - (\eta^+ - \eta^-) * \frac{i-1}{\lambda-1}}{\lambda} \tag{1.2}$$

The couple (η^-, η^+) define the slope of the linear function. However, notice that the constraint $\sum_{j=1}^{\lambda} p_i = 1$ requires that $1 \leq \eta^+ \leq 2$ and $\eta^- = 2 - \eta^+$. So Eq. 1.2 can be rewritten as in Eq. 1.3.

$$p_i = \frac{\eta^+ - 2 * (\eta^+ - 1) * \frac{i-1}{\lambda-1}}{\lambda} \tag{1.3}$$

It is common to use a value of $\eta^+ = 1.1$. This is the value that GUIDE asks as a parameter of this Selector.

3. *Deterministic Tournament*: Represented in GUIDE by the constant `TOURNAMENT`. String descriptor: "*Tournam.*"
   This selector chooses an individual by selecting a number q (≥ 2) of individuals at random and selecting the best one. q is usually 2, but can be configured in GUIDE.

4. *Stochastic Tournament.* Represented in GUIDE by the constant `STOCHASTIC_TOURNAMENT`. String descriptor: "*Stoch. Trn.*"
   This selector behaves similarly of Deterministic Tournament. Stochastic Tournament of rate R first choses **two** individuals from the population, and selects the best one with probability R (the worse one with probability $1 - R$). Real parameter R should be in $[0.5, 1]$.

5. *Random*: Represented in GUIDE by the constant `UNIFORM`. String descriptor: "*Uniform*"
   This selector is the *trivial* one: randomly selects the individuals.

6. *Sequential*: Represented in GUIDE by the constant `SEQUENTIAL`. String descriptor: "*Sequent.*"
   It performs a deterministic selection, from best to wrost individuals, in turn.

Reductors implemented in GUIDE The way of reducing a population is implemented in GUIDE under the idea of "*Reductors*". Five of them are implemented: *Sequential*, *Random*, *Deterministic Tournament*, *Stochastic*

Tournament and *Evolution Programming Tournament.* The first 4 are identical to the ones with the same name described in the Selectors explanation. The fifth (*Evolution Programming Tournament*) is described here.

Represented in GUIDE by the constant `EP_TOURNAMENT` and with a String descriptor: "*E.P. Trn.*", it proceeds by taking the union of the initial population with the intermediate one. For each individual a in this (double) population q individuals are chosen at random and compared with a, with respect with fitness values. This gives a new value for a, the rank of this tournament. Once all individuals have been assigned values, a deterministic algorithm chooses the survivors. GUIDE expects the application to specify the value of q. A typical value is 10.

Default Parameters

TO FINISH. Those parameters are NOT GOOD

1. Population size = 100
2. GUIDE assumes the problem is of type Minimization (i.e., lower fitnesses are better)
3. the criterium stop is set at 500 generations
4. 10 per cent of the initial population will go down directly to the final population (so elitism=.1)
5. 85 percent of the initial pop is fertile (so the parents are chosen from there)
6. 33 percent of the fertile population will conform the parent population. The way of choosing is by Tournament, size 2
7. Mutation rate on parents: .1
8. Crossover probability: .85
9. Offspring population is 50 per cent as big in size as the parent population
10. There is an intermediate population that is formed by the sum of 50 per cent of the offspring population, chosen by Tournament, size 2, plus 75 per cent of the non-elite population, chosen by tournament, size 2

11. Finally, the Final population is chosen with the sum of the Elite (10 percent of the initial population), plus the rest coming from the intermediate population, chosen Sequentially.

Miscellanous Details about the Evolution Engine in EO can be found on `http://www.lri.fr/~marc/EO/eo/tutorial/html/eoEngine.html`