Metaheuristic algorithms.

Lab 4: Evolution strategies. Genetic programming.

---

1. Evolution Strategies

Evolution strategies are mainly used to solve continuous optimization problems. In the case of evolution strategies, the elements of the population are real vectors and the main components are:

- *Selection:* it is used only to select the survivors (all elements can be parents) and it is usually a deterministic selection based on taking the best M offsprings from the set of L offsprings (in the case of (M,L) strategies) or the best M elements from the joined population of parents and offsprings (in the case of (M+L) variants). M denotes the number of elements in the current population and L denotes the number of elements generated using recombination and mutation.
- *Recombination:* from R parents is constructed one offspring by linear (convex) combination. For a population of M elements are constructed through recombination M new elements which are further modified by applying mutation.
- *Mutation:* it is applied to all elements in the population and consists of adding a random value (generated according to a given distribution).

Application 1. Implement a simple evolution strategy having the following characteristics:

- Convex recombination (an offspring is computed as the average of R parents – the number of parents is an input parameter)
- Mutation based on additive random perturbation relying on random values generated according to a normal distribution (N(0,sigma) = 0 mean, standard deviation equal to sigma; sigma is an input parameter)
- Selection of survivors: M+L variant based on truncation or tournament strategy.

Test functions:
See for instance:
http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

Hint: an example is implemented in SE.sci

Exercises:

1. Test SE.sci for sphere, Griewank, Ackley, Rastrigin and Rosenbrock functions described in the web page http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm
2. Analyze the impact of the parameter sigma on the ES performance by using values smaller and larger than 1.
3. Analyze the impact of the selection type ((M+L) vs. (M,L), truncation vs tournamet) on the ES performance.

Application 2.

Analyse the behaviour of the Covariance Matrix Adaptation algorithm (CMA-ES) in the case of nonseparable objective functions (the variables are correlated).

Hint. Use the CMA-ES package for Scilab (it works only for version 5.5 of Scilab) available at https://www.lri.fr/~hansen/cmaes_inmatlab.html#scilab

Steps:
(a) download the archive (from https://atoms.scilab.org/toolboxes/CMA-ES) and extract the source files in a local folder
(b) set the local folder as current in Scilab (using Change current directory)
(c) execute builder.sce and loader.sce
(d) instantiate a cmaes object using cma_new and describe the iterative process which consists of:

(i)construct a new candidate solution (using cma_ask);
(ii) update the parameters (mainly components of the covariance matrix) to be used during the nest step (using cma_tell).

Exercise:
1. Compare the results obtained using CMA-ES with those obtained using the simple evolution strategy in the case of Rosenbrock function (for n=2, 10, 100)

2. Genetic programming

The aim of genetic programming is to design in an evolutionary manner computational structures (arithmetical/logical expressions, classification/decision rules or even programs). In traditional Genetic Programming applications (as *symbolic regression*) the elements of the population are hierarchical structures (e.g. *syntactic trees*). The genetic operators are adjusted to work with such structures. One of the main difficulties in GP is to avoid the proliferation of large structures (the so called *bloat* problem). A possible solution to this problem is to limit the depth of the trees generated during the evolutionary process.

The most popular application of GP is *symbolic regression* aiming to evolve an expression which fits well to some data (unlike the numerical regression which aims to estimate the coefficients of a given model, symbolic regression estimates the model itself).

Application 3. Use the "rgp" R package to find an expression which fits a dataset.

Main steps:

- **Launch** R
- **Load package** "rgp": Packages ->Load package … or library("rgp") (if the package is not installed then it should be installed by Packages-> Install package(s)…
- **Define the set of nonterminals** (operators and functions) using functionSet.
  Example: setNonterminals <- functionSet("+", "*", "-","/")
- **Define the set of variables using** inputVariableSet.
  Example: setVariables <- inputVariableSet("x")

- **Define the set of constants** using constantFactorySet.
  Example: setConstants <- constantFactorySet(function() rnorm(1)) (random values generated using the standard normal distribution)
- **Define the test data**: values which will be used to evaluate the approximation accuracy.
  Example: dateX <- seq(from = -pi, to = pi, by = 0.1)
- **Define the fitness function**: mean square error (measure of the difference between the values of the test function and the values corresponding to the evolved expressions).
  Example: fitness <- function(f) rmse(f(dateX), sin(dateX)) (if the reference function is sinus)
- **Call the function** corresponding to the evolutionary process (geneticProgramming).
  Example:

```
geneticProgramming(functionSet = setNonterminals,
                   inputVariables = setVariables,
                   constantSet = setConstants,
                   fitnessFunction = fitness,
                   stopCondition = makeStepsStopCondition(10000))
```

Particularities of the genetic programming implemented in "rgp":

- The population elements are R expressions (implemented as tree-like structures)
- The population initialization is based on several construction strategies:
  - "grow" (each branch in the tree will be extended until it reaches the maximal length or until a random event occurs)
  - „full" (all branches in the tree have the maximal length)
  - Combined variant (some elements are generated using the "grow" strategy, others are constructed using the „full" strategy)
- The package implements the traditional crossover and mutation strategies adapted for trees (see slides of lecture 6)
- There are implemented various selection variants using one or several criteria (as in multiobjective optimization). In the multi-criteria variant the aim is to optimize the quality of the result, the simplicity of the elements and the population diversity.

Exercise 2: Follow the above steps and test the influence of nonterminals on the quality of the results (by changing the elements of the nonterminals set).
Hint: see for instance SymbolicRegression_GP.r

Homework:

1. Extend SE.sci by introducing self-adaptation of the parameter s (standard deviation of the normal distribution used in the mutation step – see Lecture 5).
2. Apply genetic programming (rgp package for R) to evolve a boolean expression which corresponds to the parity function (the parity function returns 0 if it receives an even number of variables equal to 1 and it returns 1 if it receives an odd number of variables equal to 1). Hint: see rgp_introduction.pdf