Homework Module:

Programming the Basics of an Evolutionary Algorithm(EA)

Purpose: Gain experience with the basic mechanisms underlying EAs by programming them.

1 Assignment

In the programming language of your choice (although C++ or Python is recommended), implement all of the basic components of an evolutionary algorithm and briefly test them on the One-Max problem. This homework module will normally be combined with another module that involves a slightly more complex problem to solve with your EA, for example the Knight's Tour, the Short Taps puzzle or the Colonel Blotto game.

Your EA must include the following aspects:

1. A **population** of potential solutions (i.e. individuals) all represented in some low-level *genotypic* form such as binary vectors.
2. A **developmental method** for converting the genotypes into phenotypes. For this general EA, a simple routine for converting a binary genotype into a list of integers will suffice. This can be extended for future homework modules.
3. A **fitness evaluation** method that can be applied to all phenotypes. You will want to make this a very modular component of your EA such that a wide variety of fitness functions can be experimented with.
4. All 3 of the basic **selection protocols** for adult selection described in the ea-selection.pdf chapter of the lecture notes. It is sufficient to only implement these for adult selection, not for parent selection.
5. A set (of at least 4) **selection mechanisms** (also described in ea-selection.pdf) for parent selection. These 4 must include fitness-proportionate, sigma-scaling, and tournament selection. It is sufficient to only implement these for parent selection, not for adult selection.
6. The **genetic operators** of mutation and crossover. For this generic EA, you only need to define them for binary genomes (i.e. bit vectors).
7. A basic **evolutionary loop** code for running the GA through many generations of evolution.

8. A **plotting routine** that allows the user to visualize the results of an EA run. The key visual aid is a plot of the fitness progression, showing the maximum and average fitness (along with the standard deviation) for each generation of the run. You may decide to dump the data to file and plot the fitness progression afterwards in another program, such as Python's matplotlib, Matlab, Excel or Gnuplot. There is no need to write the plot routine from scratch. Plots of fitness progression will be a standard deliverable for all homework modules involving EAs.

Aside from the plotting routine, all of the above aspects must be coded from scratch. If you are in doubt as to whether a given routine is *low-level* enough to be merely reused (instead of implemented by you), then consult the course instructor.

Your program should be highly parameterizable, so that factors such as population size, number of generations, crossover and mutation rates, etc. can be specified by the user at run-time. You should NOT need to recompile or reload your system merely to run with a different setting of any of these parameters: they should be inputs of some form, whether from the command line, an input script, or a GUI.

Your program should be object-oriented, with basic classes for populations, genotypes, phenotypes, selection mechanisms, etc. Different problems should all be handled by the same basic classes, although you will often need special subclasses of the genotypes and phenotypes for problem-specific representations. The genetic operators for these subclasses will often be different than those for the generic genotype class. However, many problems can reuse a) the binary genome, b) the mutation and crossover operators for binary genomes, and c) the conversion routines from binary segments to integers.

Before applying your EA to another homework-module problem, verify that it works by running it on the simple One-Max problem, which uses a binary genotype that translates into a binary phenotype (via a direct copy) whose fitness is simply the number of 1's that it contains. Make sure that your EA can solve One-Max problems of size 40 (bits) before moving on to other problems.

2 Deliverables

1. A clear, concise description of your EA code in text and a few diagrams **(2 points)**.
2. A justification of your code's modularity and reusability. You should describe how easy it is for your code to incorporate new phenotypes, genotypes, genetic operators and selection mechanisms as may be needed to handle new problems. **(1 point)**.
3. An analysis of the performance of your EA on a 40-bit One-Max problem where the adult selection protocol is full-generational replacement and the parent selection mechanism is fitness-proportionate. First run the problem using various population sizes until you find the approximately minimal size that allows One-Max solutions to be consistently found in under 100 generations. Then, using only that population size, experiment with different values for the crossover and mutation rates. Use fitness plots to show the different results. Make a statement as to what the best choices are for these parameters in your runs. **(2 points)**
4. Using the best-found population-size, mutation and crossover settings from the previous experiments, do a new set of experiments to find the parent selection mechanism that gives the best results. Again, document your experiments with fitness plots. **(2 points)**.
5. Modify the target bit string from all 1's to a random bit vector of length 40. Do you expect this to increase the difficulty of the problem? Run the EA and find out. Document your results from at least 4 different runs as part of your comparison. Explain the results. **(2 points)**.

Your report should be 5-10 pages long, including fitness plots and other diagrams.

Remember: the core of your program should be reusable on other evolutionary-computation homework modules!

2.1 Warning

Depending upon the actual course and semester - your instructor uses this module in various courses - you may or may not be required to do any or all of the following:

1. Demonstrate this module to the instructor or a teaching assistant.
2. Upload the report and/or code for this module to a particular site such as *It's Learning*.

Consult your course web pages for the requirements that apply.