NAT-DL: Self-study questions on Genetic Programming Set 3 (week 5)

1. Genetic programming (GP) is an evolutionary technique which attempts to evolve programs fit for some purpose. Describe a typical GP system: Explain how programs are represented in the system; Give examples of the genetic operators applied; and state the main steps of the evolutionary algorithm indicating where there are design choices to make.
2. Express the following functions in tree notation, using only “+”, “−”, “*”, “/” as non-terminals and *x*, 0, 1, 2, 3, . . . as terminals.

   a) $y = 3x + 2x$

   b) $y = 5x^4 - 2x^2$

   c) $y = -0.25x^3 + 3.5$

   Which of these functions can you represent using only *x* and 1 as terminals?
3. As it could be a bit more difficult to program a GP (compared to PSO or GA), you can start this computer exercise a notebook that is already available. It features the function x*x + 2*x + 1 which is to be reconstructed from a data set. The GO should have no problems with this simple function. What do you observe for a slightly more challenging function, such as (x+1)^3?
4. What fitness function would GP use for solving symbolic regression problems? Can you think of any alternatives? How much domain specific knowledge about the problem is encoded in this fitness function?
5. Do local minima exist in Genetic Programming? What about "building blocks"?
6. A division can be represented by a power series of the form $1/(1-x) = 1+x+x^2+ ...$ How would you implement a GP to learn to represent the right-hand side (i.e. without using a division operation)?
7. Genetic programming use parse trees to encode solutions. Defining a distance between trees is not a trivial task, but may be useful to keep track of and control the diversity in the population. Propose such a distance to deal with tree-based representations. [Adapted from E. Talbi: Metaheuristics]
8. Even-parity-4 problem using GP. In its general formulation, the Boolean even-parity *k* function of *k* Boolean arguments returns true if an even number of its Boolean arguments evaluates true, otherwise it returns false. For the special case where *k* = 4, 16 fitness cases must be checked to evaluate the fitness of an individual. Formulate the objective function for this problem. Our goal is to design a genetic programming approach to solve the problem. Propose a tree-like representation for this problem. For this purpose, the operators and the terminal set of the tree have to be defined. [From E. Talbi: Metaheuristics]