Computational Statistics
Combinatorial optimization

The data `baseball.dat` from the book "Computational statistics" by Givens and Hoeting contains salaries in 1992 and 27 performance statistics for 337 baseball players in 1991 (see the file `baseball.txt` for a description of the data). The objective of the study is to find the subset of variables that best explains the salary using a linear regression model.

1. Using function `read.table`, store the data as a data table. Display matrix plots of part of the data (use function `plot`).

2. We wish to program a *steepest descent algorithm* that finds the linear model minimizing the AIC criterion for this data. For this, we will proceed in several steps:

   (a) Write a function `initialize` than generates an random initial solution. For this, you will have to propose a way of coding a solution (i.e., a subset of predictors). For instance, it can be coded as a vector `theta` of length $p = 27$ such that `theta[i]=1` if predictor i is present and `theta[i]=0` otherwise.

   (b) Write a function `aic` that computes the AIC criteria of a solution `theta`. (Use the built-in functions `lm` and `AIC`).

   (c) Write a function `neighborhood` that computes all the neighbors of a solution (i.e., all the alternative solutions obtained by adding or removing exactly one predictor).

   (d) Write a function `local_search` that implements the steepest descent algorithm.

3. We now wish to program the *simulated annealing algorithm*. For this, we will again proceed in several steps:

   (a) Write a function `new` that draws randomly an alternative solution from the current one, by adding or removing exactly one predictor.

(b) Write a function `simulated_annealing` that implements the simulated annealing algorithm, with the following cooling schedule:

$$\tau_{j+1} = a\tau_j$$
$$m_{j+1} = m_j + b,$$

where τ_j is the temperature at stage j, m_j the number of iterations at stage j, and a and b are user-defined coefficients. Stop the algorithm when the temperature becomes less that some threshold `taumin`.

(c) Experiment with different values of a and b.

4. We will now apply a genetic algorithm to our problem. For this, we will use the package `GA`

(a) Install the package `GA` using the command `install.packages("GA")`, and load it using the command `library("GA")`. We will use the function `ga`.

(b) Write the fitness function. It should be almost similar to the function `aic` above, except that the objective function should now be *maximized*.

(c) Experiment with the function `ga`. In particular, vary the population size and mutation probability. Display the results with the function `plot`.