Evolutionary Algorithms
Problem Set - MATLAB

General
This problem set contains MATLAB exercises to get you acquainted with the use of MATLAB for Evolutionary Algorithms. The exercises are not mandatory, but are very useful for your understanding of the subject matter, and in completing the practical assignment. If you have questions or want to discuss your solutions, contact the teaching assistants.

1 Counting Ones
The simplest binary test problem for Evolutionary Algorithms is the Counting Ones problem:

Input: binary sequences $\vec{a} \in \{0, 1\}^n$.

Objective function:

$$f(\vec{a}) = \sum_{i=1}^{n} a_i.$$  \hspace{1cm} (1)

Implement the objective function in MATLAB.

2 Low Autocorrelation of Binary Sequences
The Low Autocorrelation of Binary Sequences problem is described by:

Input: binary sequences $\vec{a} \in \{-1, +1\}^n$.

Objective function:

$$f(\vec{a}) = \frac{n^2}{2 \cdot E(\vec{a})},$$  \hspace{1cm} (2)

$$E(\vec{a}) = \sum_{k=1}^{n-1} \left(\sum_{i=1}^{n-k} a_i \cdot a_{i+k}\right)^2.$$  \hspace{1cm} (3)

Implement the objective function in MATLAB.

3 Ackley
The Ackley function is a frequently used real-valued test function for Evolutionary Algorithms:

Input: vectors of reals $\vec{x} \in \mathbb{R}^n$.

Objective function:

$$A_{c_1, c_2, c_3}(\vec{x}) = -c_1 \cdot \exp \left( -c_2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos(c_3 \cdot x_i) \right) + c_1 + e.$$  \hspace{1cm} (4)

Implement the Ackley function in MATLAB.

Plot the function in 3D (i.e., $\vec{x} = (x_1, x_2)$, plotted against the function value $A_{c_1, c_2, c_3}(\vec{x})$) with $x_i \in [-5, 5]$, and $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2\pi$. Hint: use the commands `meshgrid` and `surfc`. 
4 Fletcher and Powell

The function after Fletcher and Powell, an example of a nonlinear parameter estimation problem, is described by:

**Input**: vectors of reals $\vec{x} \in \mathbb{R}^n$.

**Objective function**:

$$F_{A,B,\vec{\alpha}}(\vec{x}) = \sum_{i=1}^{n} (p_i(A,B,\vec{\alpha}) - q_i(A,B,\vec{x}))^2,$$

where $A$ and $B$ are $n \times n$ matrices (i.e., $A$ and $B$ contain $n^2$ elements $a_{ij}$ and $b_{ij}$ respectively) and $A$, $B$, and $\vec{\alpha}$ consist of random real values.

Implement this function in MATLAB, assuming $A$, $B$, $\vec{\alpha}$, and $\vec{x}$ are given as input.

5 Monte-Carlo Search for Binary Problems

Implement a Monte-Carlo Search algorithm (see slide 14 of lecture 1) for maximization problems in MATLAB, in order to solve the Counting Ones problem. Let the function take the following arguments:

- the objective function;
- the length $n$ of the bitstrings $\vec{a}$;
- the maximum number of fitness function evaluations.

Let the function produce as outputs:

- the best individual found by the algorithm;
- the fitness of the best individual.

Use $n = 100$ and for a run of 100 iterations, plot the fitness against the elapsed number of iterations.

Hint: store the current best solution found so far, and compare fitness value of new solutions with this current best solution. Furthermore, you can pass a function as argument by putting the `@`-symbol before the function name (i.e., it is passed as function handle).

6 Simple (1+1)-GA for Binary Problems

Implement the alternative method given in slide 18 of lecture 1 (this can be regarded as a simple (1+1)-GA) for solving the Counting Ones problem, using the same approach as with the Monte-Carlo Search algorithm of assignment 5.

Use $n = 100$ and a mutation rate $p = 1/n$. For a run of 100 iterations, plot the fitness against the elapsed number of iterations. Is there a difference in performance compared to the Monte-Carlo Search algorithm?
7 Monte-Carlo Search for Real-Valued Problems

Implement a Monte-Carlo Search algorithm for real-valued minimization problems in MATLAB. Let the function take the following arguments:

- the objective function;
- the length $n$ of the vector $\vec{x}$;
- a vector of length $n$ containing the lower bounds of the search domain;
- a vector of length $n$ containing the upper bounds of the search domain;
- the maximum number of fitness function evaluations.

Let the function produce as outputs:

- the best individual found by the algorithm;
- the fitness of the best individual.

Run the algorithm on the Ackley function for 100 iterations with $n = 2$, $x_i \in [-5, 5]$, and $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2\pi$. Plot the fitness versus the number of iterations.

Note: we are now optimizing a minimization problem.

8 Simple (1+1)-EA for Real-Valued Problems

Think of a way of implementing the alternative method of assignment 6, now for dealing with real-valued optimization problems.