Metaheuristic algorithms.

Lab 7:
Multimodal optimization
Evolutionary training of neural networks

---

1. Multimodal optimization

Depending on their goal, the optimization problems can belong to one of the following categories:

- *Global optimization:* the aim is to find a configuration which minimizes or maximizes the value of the objective function (among all possible configurations in the search space).
- *Local optimization:* the aim is to find the best configuration in the neighborhood of a given (initial) element. A local optimum is better than the elements in its neighborhood but it could be worse than the global optimum.
- *Multimodal optimization:* the aim is to find all optima (both local and global); it is useful when there are several global optima and/or the local optima correspond to interesting configurations. The interest in identifying all optima might appear in engineering design (e.g. find all resonance points of a mechanical or electrical system).

The identification of all optima can be done in at least two ways:

- By iterating a local optimization algorithm (starting from different initial configurations)
- By using a single run of a population-based algorithm which enforce the population to discover several niches in the search space, each niche being related to an optimum.

The global optimization metaheuristics favor the populations which converges in a neighborhood of the optimum, such that at the end of the iterative process the population is characterized by a small diversity. In the case of multimodal optimization the strategy should be different, as the population should re-organize itself in species associated to the regions of all optima. Such a speciation should be achieved through explicit division of population in sub-populations (which is not effective if the number of optima is unknown) or through implicit speciation (the elements group themselves in species as an effect of some specific mechanisms). The most popular speciation mechanisms are:

- *Sharing*: the fitness of each element is penalized (by using a sharing function) if it belongs to a crowded region.
- *Crowding*: the main idea is that during the selection process an element competes only with elemente which are in its neighborhood (e.g. with the closest element)

Example: Crowding DE (proposed by R. Thomsen „Multimodal optimization using crowding-based differential evolution", CEC 2004) is different from a standard DE (see lab 5) only with respect to the selection step. In the standard DE, each trial element (z_i)is compared only with the corresponding current element (x_i). In crowding DE the trial element replaces the element from the current population which is the closest and it replaces it if it is better.

Fig 1 illustrates the effect of this change in the DE selection.

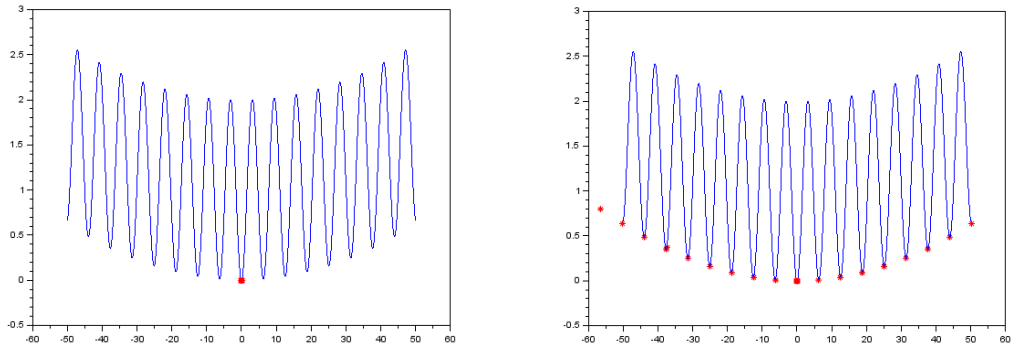

Fig. 1. Distribution of a population with 30 elements after 500 generation. Standard DE (left) and crowding DE (right)

Application 1. Modify the implementation of the DE algorithm (lab 5) in order to incorporate the crowding-based selection and test for one-dimensional and 2-dimensional multimodal test functions. Hint: see DEcrowding.sci

2. Evolutionary training of neural networks

Evolutionary algorithms (but also other population-based metaheuristics) can be used for neural networks design at least in two cases:

- Estimation of the synaptic weights in the case of networks with non-differentiable activation/transfer functions or in the case of recurrent networks (when traditional algorithms like Backpropagation cannot be applied)
- Establishing the architecture of the network (both in the case of explicit and in the case of implicit encoding – see Lecture 13-14) .

Application 2. Let us consider the problem of training a neural network in order to represent the XOR function.

We can use the following architecture:

- 2 input units + 1 dummy unit (used to provide the biases for the hidden units)
- K hidden units (K is an input parameter) with a Heaviside activation function + 1 dummy unit (used to provide the bias for the output unit)
- An output unit with a Heaviside activation function

Each element of the population will have 4*K+1 components corresponding to all weights and biases. The function to be minimized is the MSE (Mean Squared Error) computed for the training set containing all four examples: ((0,0),0), ((0,1),1),((1,0),1), ((1,1),0). Analyze the influence of the population sizes, mutation parameter, number of parents used for recombination, selection type (truncation or tournament) and value of K.

Hint. see SE_nn.sci. The function used by the evolution strategy to evaluate a network is SE_XOR.

Exercise: Use DE (differential evolution) or PSO (particle swarm optimization) instead of the evolution strategy to estimate the parameters of the neural network.

Homework. Implement an evolutionary approach which estimate simultaneously both the number of hidden units and the weights